2/5/1 (Item 1 from file: 351)
DIALOG(R)File 351:Derwent WPI

011830033 **Image available**
WPI Acc No: 1998-246943/**199822**
XRPX Acc No: N98-195633
Simultaneous instruction-execution method of microprocessor - involves copying contents of registration file for main instruction, which starts new instruction execution, to registration file for new instruction via content-batch forwarding unit based on copy request
Patent Assignee: NEC CORP (NIDE)
Inventor: TORII S
Number of Countries: 002 Number of Patents: 003
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
JP 10078880	A	19980324	JP 96249272	A	19960830	199822	B
US 5913059	A	19990615	US 97900643	A	19970725	199930	
JP 2970553	B2	19991102	JP 96249272	A	19960830	199951	

Priority Applications (No Type Date): JP 96249272 A 19960830
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
JP 10078880	A		32	G06F-009/46	
JP 2970553	B2		32	G06F-009/46	Previous Publ. patent JP 10078880
US 5913059	A			G06F-009/40	

Abstract (Basic): JP 10078880 A
The method involves using several logical program counters as basis for the simultaneous execution of various instructions including e.g. fetch, interpretation. The execution of new instruction is started based on the instruction designated in a single instruction.
The contents of the registration file (6a,6b) of the instruction execution point of main instruction, which starts the execution of new instruction, is copied to another registration file (6b) for new instruction via a content-batch forwarding unit (7) based on a copy request.
ADVANTAGE - Reduces overhead of parallel processing by efficient inheritance of instruction contents of one registration file to another registration file, thus improving processing speed when performing instruction level parallel processing.
Dwg.2/33
Title Terms: SIMULTANEOUS; INSTRUCTION; EXECUTE; METHOD; MICROPROCESSOR; COPY; CONTENT; REGISTER; FILE; MAIN; INSTRUCTION; START; NEW; INSTRUCTION ; EXECUTE; REGISTER; FILE; NEW; INSTRUCTION; CONTENT; BATCH; FORWARDING; UNIT; BASED; COPY; REQUEST
Derwent Class: T01
International Patent Class (Main): G06F-009/40; G06F-009/46
International Patent Class (Additional): G06F-009/38; G06F-015/16; G06F-015/177
File Segment: EPI

2/5/2 (Item 1 from file: 347)
DIALOG(R)File 347:JAPIO

05795780 **Image available**
METHOD FOR EXECUTING MULTI-THREAD

PUB. NO.: 10-078880 A]
PUBLISHED: March 24, 1998 (19980324)

INVENTOR(s): TORII ATSUSHI
APPLICANT(s): NEC CORP [000423] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 08-249272 [JP 96249272]
FILED: August 30, 1996 (19960830)
INTL CLASS: [6] G06F-009/46; G06F-009/38; G06F-015/16
JAPIO CLASS: 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units); 45.4 (INFORMATION PROCESSING -- Computer Applications)
JAPIO KEYWORD: R131 (INFORMATION PROCESSING -- Microcomputers & Microprocessers)

ABSTRACT

PROBLEM TO BE SOLVED: To reduce the overhead of a parallel processing, and to attain a parallel processing with high performance by allowing a newly generated thread to efficiently succeed the content or a register at the time of operating the parallel processing of a thread level.

SOLUTION: A sled executing means 5a executes a fork instruction, and makes a thread generating request to a thread managing means 4. The thread managing means 4 retrieves a sled executing means 5b in an execution waiting state, and makes a thread activating request to this. At the same time, the sled managing means 4 makes the copy request of a register content from a register file 6a to a register file 6b to a register content batch transferring means 7. The register content batch transferring means 7 operates the copy of the register content from the register file 6a to the register file 6b according to this request.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－78880

(43)公開日　平成10年(1998)３月24日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	9/46	３６０		Ｇ０６Ｆ	9/46	３６０Ｂ
	9/38	３７０			9/38	３７０Ａ
	15/16	４３０			15/16	４３０Ｂ

審査請求　有　　請求項の数11　ＦＤ　（全 32 頁）

(21)出願番号　特願平8－249272

(22)出願日　平成８年(1996)８月30日

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号

(72)発明者　鳥居　淳
東京都港区芝五丁目７番１号　日本電気株式会社内

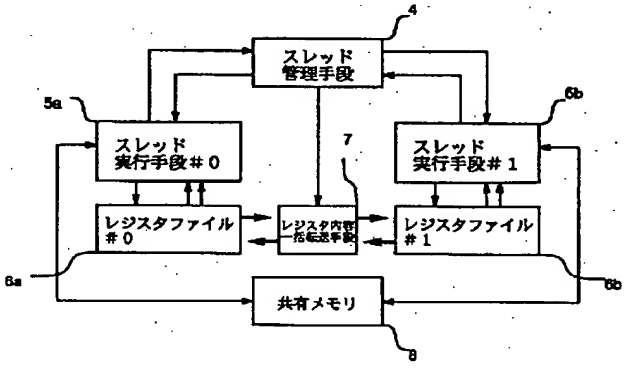
(74)代理人　弁理士　境　廣巳

(54)【発明の名称】　マルチスレッド実行方法

(57)【要約】

【課題】　スレッドレベルの並列処理を行う際に、新規生成スレッドに効率的にレジスタの内容を継承させることによって、並列処理のオーバヘッドを削減し、高性能な並列処理を可能にする。

【解決手段】　スレッド実行手段５ａはフォーク命令を実行すると、スレッド管理手段４に対してスレッド生成要求を行う。スレッド管理手段４は実行待ち状態のスレッド実行手段５ｂを検索し、これに対してスレッド起動要求を行う。同時に、スレッド管理手段４は、レジスタ内容一括転送手段７に対してレジスタファイル６ａからレジスタファイル６ｂへのレジスタ内容のコピー要求を行う。レジスタ内容一括転送手段７は、この要求に従いレジスタファイル６ａからレジスタファイル６ｂへのレジスタ内容のコピーを行う。



1

【特許請求の範囲】

【請求項1】　論理的に複数のプログラムカウンタを有し、これらのプログラムカウンタに従った複数のスレッドの命令を同時にフェッチ，解釈，実行するスレッド実行手段と、独立した論理的なレジスタファイルとを備え、単一のスレッド中の指定された命令によって、新たなスレッド（子スレッド）の実行を開始する機能を備えたプロセッサにおいて、

子スレッド実行開始命令を実行するスレッド（親スレッド）の命令実行時点のレジスタファイルの内容を子スレッドのレジスタファイルに継承することを特徴としたマルチスレッド実行方法。

【請求項2】　スレッド毎に複数のレジスタファイル及びレジスタ選択手段を備えると共に、レジスタ内容転送手段を備え、

子スレッドは、前記レジスタ選択手段によって親スレッドのレジスタファイルを参照し、

レジスタの変更が行われる毎に、前記レジスタ内容転送手段によって順次新スレッドのレジスタファイルに、変更する前のレジスタ内容を転送し、

前記レジスタ選択手段の選択内容を子スレッド側のレジスタファイルに切替えることによってレジスタファイルを継承させることを特徴とした請求項1記載のマルチスレッド実行方法。

【請求項3】　親スレッドのプログラムにおける子スレッド実行開始命令とその前後に配置される他の演算命令との間で、プログラムの意味を変えない範囲内において非プログラム順序に実行することを特徴とした請求項1記載のマルチスレッド実行方法。

【請求項4】　演算の結果とレジスタ番号を指定するタグを一時的に格納するリオーダバッファを持ち、命令をデコードした際に、そのリオーダバッファのエントリを確保し、命令をプログラムで指定された順序ではなく、必要なレジスタの値が使用可能になったものからレジスタ及びリオーダバッファの当該命令よりも前のエントリから供給することによって演算を行い、演算結果を命令デコード時に確保したリオーダバッファのエントリに格納し、そのリオーダバッファからはプログラム順序でレジスタ内容の更新を行うことによって、プログラムの順序に従わずに処理を進める非プログラム順序実行型プロセッサにおいて、

複数のプログラムカウンタを有し、これらのプログラムカウンタに従った複数のスレッドの命令を同時にフェッチ，解釈，実行するスレッド実行手段と、複数のレジスタファイルおよびリオーダバッファを設け、

親スレッドと子スレッドの両者のレジスタファイルおよびリオーダバッファからの内容出力を選択するレジスタデータセレクタ装置を設け、

子スレッド生成命令が親スレッド内でプログラム順序で完了した時点で、親スレッドを実行しているレジスタファイルの内容を子スレッドを実行しているレジスタファイルにコピーすることによって、

コピー前は親スレッドのレジスタファイル，リオーダバッファおよび子スレッドのリオーダバッファからレジスタ内容を選択し、コピー後は子スレッドのリオーダバッファおよびレジスタファイルからレジスタ内容を選択することによってレジスタの継承を行うことを特徴とした請求項3記載のマルチスレッド実行方法。

【請求項5】　演算の結果とレジスタ番号を指定するタグを一時的に格納するリオーダバッファを持ち、命令をデコードした際に、そのリオーダバッファのエントリを確保し、命令をプログラムで指定された順序ではなく、必要なレジスタの値が使用可能になったものからレジスタ及びリオーダバッファの当該命令よりも前のエントリから供給することによって演算を行い、演算結果を命令デコード時に確保したリオーダバッファのエントリに格納し、そのリオーダバッファからはプログラム順序でレジスタ内容の更新を行うことによって、プログラムの順序に従わずに処理を進める非プログラム順序実行型プロセッサにおいて、

複数のプログラムカウンタを有し、これらのプログラムカウンタに従った複数のスレッドの命令を同時にフェッチ，解釈，実行するスレッド実行手段と、複数のレジスタファイルおよびリオーダバッファを設け、

親スレッドと子スレッドの両者のリオーダバッファ及び子スレッドのレジスタファイルからの内容出力を選択するレジスタデータセレクタ装置を設け、

レジスタファイルの内容のコピーを子スレッド生成が行われた時点で行うこととして、その後は親スレッドのリオーダバッファから、親スレッドのレジスタファイルと子スレッドのレジスタファイルに書き込みを行い、

スレッド生成命令がプログラム順序で終了する前は親スレッドのリオーダバッファ，子スレッドのレジスタファイルおよびリオーダバッファからレジスタ内容を選択し、プログラム順序で終了した後は子スレッドのリオーダバッファおよびレジスタファイルからレジスタ内容を選択することによってレジスタの継承を行うことを特徴した請求項3記載のマルチスレッド実行方法。

【請求項6】　レジスタファイルのコピーを複数回の転送サイクルによって行うレジスタ内容転送手段を用い、レジスタ内容の転送が済んだレジスタファイル部分から、新スレッドにおいて、参照を許可することを特徴とした請求項4または5記載のマルチスレッド実行方法。

【請求項7】　スレッド生成命令を実行した時点で、そのスレッドを実行できる資源が確保できない場合に、レジスタの内容を退避用レジスタファイルに蓄えることによって、プロセッサの許容数以上のスレッドが同時に存在し得るようにしたことを特徴とする請求項4，5または6記載のマルチスレッド実行方法。

【請求項8】　プログラムから指定する論理レジスタと

ハードウェアに実行される物理レジスタとの対応関係を可変とし、この対応関係を記録，更新，検索するレジスタ写像テーブルを備え、論理レジスタに対して値の書き込みを行おうとする毎に物理レジスタとの新しい対応関係を生成し、対応する物理レジスタに値の書き込みが完了した時点で読み出しを許可し、命令がプログラム順序に完了した時点で、論理レジスタとの古い対応関係になっていた物理レジスタを未使用状態にすることによって、非プログラム順序で命令を実行するプロセッサにおいて、

複数のプログラムカウンタを有し、これらのプログラムカウンタに従った複数のスレッドの命令を同時にファイル，解釈，実行するスレッド実行手段を設け、

前記複数のスレッド実行手段から参照できる共有の物理レジスタファイルと、この共有の物理レジスタファイルの使用状態保持手段と、複数のレジスタ写像テーブルとを設け、

子スレッド実行開始命令を解釈した時点で、これらのレジスタ写像テーブル間で内容をコピーすることによってレジスタの継承を行い、

継承した物理レジスタを親スレッド，子スレッド両者で対応する論理レジスタに書き込みを行った時点で、未使用状態にすることによってレジスタの継承を行うことを特徴とした請求項3記載のマルチスレッド実行方法。

【請求項９】　レジスタ写像テーブルの内容のコピーを複数回の転送サイクルによって行い、

新スレッドにおいて、対応関係の転送が済んだものから使用可能にすることを特徴とした請求項８記載のマルチスレッド実行方法。

【請求項１０】　スレッド生成命令を実行した時点で、そのスレッドを実行できる資源が確保できない場合に、レジスタ写像テーブルの内容を退避用レジスタ写像テーブルに蓄えることにより、

プロセッサの許容数以上のスレッドが同時に存在し得るようにすることを特徴とした請求項８または９記載のマルチスレッド実行方法。

【請求項１１】　親スレッドから子スレッドを生成する回数を高々１回に制限し、スレッドの消滅順序を親スレッドを子スレッドより先にすることによって、

隣接するスレッド実行手段に限定してレジスタ内容継承を行うようにしたことを特徴とする請求項２，３，４，５，６，７，８，９または１０記載のマルチスレッド実行方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は複数の命令を同時に実行する高性能なマイクロプロセッサに関し、特にマルチスレッド実行方法に関する。

【０００２】

【従来の技術】プロセッサの高速化の技術として、複数

の演算装置を用意し、命令単位の並列性を利用して複数の演算装置に同時に命令を発行することにより処理速度を向上る方式が実用化されている。この方式を用いた場合、理想的には１クロックで複数の命令を実行することが可能である。

【０００３】しかしながら、現実には命令間に依存関係が存在し、前の命令が終了しないと後ろの命令が実行できない場合が存在するため、同時に実行できる命令数は制限されてしまう。また、条件分岐命令によって、命令がスムーズに演算装置に供給できないという現象も生じる。これらのことから、演算装置を無限に増やした場合にも、実際には３～４倍の性能向上に抑えられてしまうといわれている。

【０００４】このような性能向上の限界は、Ｍｏｎｉｃａ　Ｓ．Ｌａｍ氏らが１９９２年に発表した論文（Ｍｏｎｉｃａ　Ｓ．Ｌａｍ　ａｎｄ　Ｒｏｂｅｒｔ　Ｐ．Ｗｉｌｓｎ，”Ｌｉｍｉｔｓ　ｏｆ　Ｃｏｎｔｒｏｌ　Ｆｌｏｗ　ｏｎ　Ｐａｒａｌｌｅｌｓｉｍ”，ＩＴｈｅ　１９ｔｈ　Ｉｎｔｅｒｎａｔｉｏｎａｌ　Ｓｙｍｐｏｓｉｕｍ　ｏｎ　Ｃｏｍｐｕｔｅｒ　Ａｒｃｈｉｔｅｃｔｕｒｅ，ＩＥＥＥ　Ｃｏｍｐｕｔｅｒ　Ｓｏｃｉｅｔｙ　Ｐｒｅｓｓ，１９９２，ｐｐ．４６－５７）に述べられている。

【０００５】この限界を踏まえ、さらなる性能向上を図る手段として、

○命令単位の並列処理をさらに進めるために、非プログラム順序実行機構（ｏｕｔ－ｏｆ－ｏｒｄｅｒ実行機構）やレジスタリネーミング機構を導入し、命令間の依存関係を少なくする。

○プログラムを複数の命令流（スレッド）に分割して、そのスレッドレベルで並列処理を行う。などの技術が提案されている。

【０００６】最初の項目の、ｏｕｔ－ｏｆ－ｏｒｄｅｒ実行機構はプログラムの実行順序に従わずに実行可能になった命令から先に実行するための機構である。このためには、レジスタ間でデータの直接の依存関係はないが、レジスタ割り付けの際にレジスタ数不足によって生じるような偽りの依存関係を解消する必要がある。このような依存に対して、ソフトウェアで指定されたレジスタの名前に対し、ハードウェアによって別のレジスタの名前に写し変えることを行うのが、レジスタリネーミング機構である。

【０００７】例えば、以下のプログラム順序で命令が与えられているとする。

```
１０　　ａｄｄ　　ｒ１←ｒ２＋ｒ３
１４　　ｓｕｂ　　ｒ４←ｒ１－ｒ５
１８　　ａｄｄ　　ｒ５←ｒ６＋ｒ７
１ｃ　　ｓｕｂ　　ｒ８←ｒ４－ｒ５
```

ここで、０ｘ１４番地のｓｕｂ命令は、０ｘ１０番地のａｄｄ命令の結果（ｒ１レジスタ）を用いているので、

直接のデータ依存関係があることになる。同様に０ｘ１ｃ番地のｓｕｂ命令は、ｒ５レジスタを介して０ｘ１８番地のａｄｄ命令と依存関係がある。しかしながら、０ｘ１４番地のｓｕｂ命令で読み出し参照され、０ｘ１８番地のａｄｄ命令で書き込み参照されるｒ５レジスタには、データの依存関係は存在しないが、レジスタの再使用による偽りの依存関係が存在する。従って、ｏｕｔ－ｏｆ－ｏｒｄｅｒ実行機構を備えるだけでは、０ｘ１８番地のａｄｄ命令は０ｘ１４番地のｓｕｂ命令を追い越すことはできない。レジスタリネーミング機構は、このような場合に、０ｘ１４番地のｒ５と０ｘ１８番地のｒ５を異なったレジスタにリネーミングすることによって、ｏｕｔ－ｏｆ－ｏｒｄｅｒ実行を可能とするものである。

【０００８】このレジスタリネーミング機構は、リオーダバッファ方式とレジスタ写像テーブル方式の２種類に大別される。図３０にリオーダバッファ方式の構成図を、図３１にレジスタ写像テーブル方式の構成図を示す。

【０００９】リオーダバッファ方式は、ｏｕｔ－ｏｆ－ｏｒｄｅｒで計算された演算結果をすべてリオーダ・バッファ１２３と呼ばれる命令毎にエントリが確保されるバッファに一旦格納し、このリオーダ・バッファ１２３からプログラム順序でレジスタファイル１２２に書き戻しを行う。後続命令はリオーダ・バッファ１２３からの値をレジスタファイル１２２からの値に優先して用いることによって、ｏｕｔ－ｏｆ－ｏｒｄｅｒ実行のレジスタリネーミングを実現する。

【００１０】また、レジスタ写像テーブル方式は、プログラムから指定する論理レジスタ番号を物理レジスタ番号にレジスタ写像テーブル１２４によって変換して、レジスタリネーミングを行う。レジスタに書き込みが生じる度に、レジスタフリーテーブル１２８から指定された未使用の物理レジスタを論理レジスタに割り付ける。また、有効命令順序バッファ１２６によって命令がプログラム順序で終了する度に、その命令が新たに生成する前の古い写像関係にある物理レジスタを未使用状態にする。このように物理レジスタを使い回すことによって、レジスタリネーミングを実現する。

【００１１】このような機構を用いた例として、リオーダバッファ方式は米Ｉｎｔｅｌ社のＰｅｎｔｉｕｍ－Ｐｒｏプロセッサ、レジスタ写像テーブル方式は米ＭＩＰＳＴｅｃｈｎｏｌｏｇｙ　ＩＮＣ．社のＰ１００００プロセッサなどの例があげられる。

【００１２】しかしながら、このような機構を用いた場合にもｏｕｔ－ｏｆ－ｏｒｄｅｒ実行可能な命令の範囲は最も古い命令から１６～３２命令程度であり、実行可能な命令があまり存在しないことも多い。また、この範囲を現実的な範囲で増やした場合も実行できる命令よりも依存などによって実行できない命令が多く入ることになり、ハードウェア量増加分の性能向上は見込めない。

【００１３】一方、二つ目の項目に示したスレッドレベルの並列処理方式は、命令単位の並列性ではなく、複数のスレッドの命令を並列に実行することにより演算装置の利用効率をあげて処理速度向上を図る方法である。この方法では、スレッド間では依存関係が少ないため前記の命令レベルの並列処理より性能向上が図りやすい。

【００１４】このスレッドレベルの並列処理は、

○全くスレッド間に依存関係のないもの。

○１スレッドの実行命令が多く、スレッド間の依存関係が少なく、ソフトウェアによって依存を解消しても性能上問題の少ないもの。

○１スレッドの実行命令が少なく、かつスレッド間の依存関係が多いため、ハードウェアによってスレッドレベル並列処理の実行支援が必要なもの。

に大別される。

【００１５】スレッド間で依存が全くない場合や、依存が少なくスレッド粒度の大きいものでは、ハードウェアによるサポートはほとんど必要ない。このような場合のプロセッサの並列方法の実施例としては、平田氏らが１９９３年に発表した論文（平田　博章，木村　浩三，永峰　聡，西澤　貞次，鷲島　敬之，「多重スレッド・多重命令発行を用いる要素プロセッサ・アーキテクチャ」，情報処理学会論文誌，１９９３年　Ｖｏｌ３４　Ｎｏ．４　ｐｐ．５９５－６０５）で提案された方法などがあげられる。この方法の実施例を図３２に示す。

【００１６】図３２の例は、命令取得装置１２９，命令解読装置１３０，機能実行装置１３１，命令間の依存解析装置１３２，機能実行装置１３１をスケジュールする命令調停装置１３３から構成される。命令解読装置１３０は命令調停装置１３３が命令を受け入れられる状態にあり、また、命令依存解析装置１３２から、命令発行可能である旨の指示を受けている場合に、命令を発行する。各々の命令解読装置１３０から発行された命令は、命令調停装置１３３によって、必要な機能実行装置１３１に割り当てられ、実際の実行が行われる。この命令調停装置１３３によって、機能実行装置１３１は各命令間で共有され、利用効率を向上させることが可能となる。また、命令調停装置１３３を機能実行装置１３１毎に分散することにより、命令調停装置１３３の簡単化が可能である。

【００１７】しかしながら、この方法では、スレッドの生成や演算データのスレッド間の伝達に対しては考慮されていない。従って、単一タスクを複数スレッドに分割し、そのタスクを高速化する場合に対処できるものではなかった。

【００１８】単一タスクの処理を高速化する際には、効率的なスレッド生成とスレッド間のデータの引き渡しが不可欠である。このような、細粒度スレッドの並列処理プロセッサの例として、Ｇｕｒｉｎｄｅｒ　Ｓ．Ｓｏｈ

i氏らが１９９５年に発表した論文（Ｇｕｒｉｎｄｅｒ Ｓ. Ｓｏｈｉ，Ｓｃｏｔｔ Ｅ. Ｂｒｅａｃｈ ａｎｄ Ｔ. Ｎ. Ｖｉｊａｙｋｕｍａｒ，“Ｍｕｌｔｉｓｃａｌａｒ Ｐｒｏｃｅｓｓｏｒ”，Ｔｈｅ ２２ｎｄ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｓｙｍｐｏｓｉｕｍ ｏｎ Ｃｏｍｐｕｔｅｒ Ａｒｃｈｉｔｅｃｔｕｒｅ，ＩＥＥＥ Ｃｏｍｐｕｔｅｒ Ｓｏｃｉｅｔｙ Ｐｒｅｓｓ，１９９５，ｐｐ．４１４－４２５）があげられる。

【００１９】Ｍｕｌｔｉｓｃａｌａｒ Ｐｒｏｃｅｓｓｏｒでは、単一のプログラムをいくつかの基本ブロックの集合である「タスク」に分割し、これを並列に実行処理できるプロセッサで処理する。タスク間でのレジスタ内容の引渡しは、タスク生成コンパイラによって生成されたｔａｓｋ ｄｅｓｃｒｉｐｔｏｒによって指定する。ｔａｓｋ ｄｅｓｃｒｉｐｔｏｒでは、生成される可能性のあるレジスタを明示的に指定する。この指定をｃｒｅａｔｅ ｍａｓｋと呼ぶ。また、最後にｃｒｅａｔｅ ｍａｓｋに指定したレジスタを更新する命令にはフォワードビットを付加する。このように、Ｍｕｌｔｉｓｃａｌａｒ Ｐｒｏｃｅｓｓｏｒはコンパイラ解析能力に依存したコードによって並列実行を行う。

【００２０】図３３は、このＭｕｌｔｉｓｃａｌａｒ Ｐｒｏｃｅｓｓｏｒの実施例である。Ｍｕｌｔｉｓｃａｌａｒ Ｐｒｏｃｅｓｓｏｒはシーケンサ１３４，プロセッシングユニット１３５，結合ネットワーク１３６，データバンク１３７から構成される。プロセッシングユニット１３５は命令キャッシュ１３８，実行ユニット１３９，レジスタファイル１４０から構成され、システムに複数存在する。また、対応してデータバンク１３７も複数存在し、ＡＲＢ（Ａｄｄｒｅｓｓ Ｒｅｓｏｌｕｔｉｏｎ Ｂｕｆｆｅｒ）１４１，データキャッシュ１４２から構成される。複数のタスクの同時実行の管理はシーケンサ１３４によって行われ、各プロセッシングユニット１３５にタスクを割り付ける。レジスタの内容はｔａｓｋｄｅｓｃｒｉｐｔｏｒの指定によって、データ生成時点でフォワードされる。

【００２１】しかしながら、従来コードをスレッドレベル並列処理に変換する場合や、依存解析の難しいコードに対してはＭｕｌｔｉｓｃａｌａｒ Ｐｒｏｃｅｓｓｏｒでは性能向上は図れない。また、コードサイズがｔａｓｋ ｄｅｓｃｒｉｐｔｏｒによって増加するという問題点も生じる。また、ｏｕｔ－ｏｆ－ｏｒｄｅｒ実行に対応した技術でないため、既存の命令レベル並列処理による性能向上が行えず、従来技術に比しての性能向上が限られていた。

【００２２】

【発明が解決しようとする課題】従来のスレッドレベル並列処理のスレッド生成技術では、明示的にレジスタの内容を継承させるか、メモリを経由して継承させるため、フォーク時に自動的にはレジスタ内容は引き継がれ

なかった。この為、レジスタ依存を記述するか、メモリに対するストア／ロードを用いて新規生成スレッドに対してデータを引き継ぐ必要があった。従って、スレッドを生成する時には生成側にも被生成側にもスレッド生成にともなうデータ継承のため命令を挿入する必要があった。

【００２３】また、ｏｕｔ－ｏｆ－ｏｒｄｅｒ実行型のプロセッサでは、同期命令などについてはその正当性を保つためにｉｎ－ｏｒｄｅｒ実行を行っているが、この場合の性能低下は顕著であった。従って、フォーク命令によってスレッドを生成し、細粒度スレッドの場合でも処理速度向上を目指す場合には、フォーク命令前と後の命令間でもｏｕｔ－ｏｆ－ｏｒｄｅｒ実行する必要があった。

【００２４】本発明の目的はこのようなスレッドレベルの並列処理を行う際に、新規生成スレッドに効率的にレジスタの内容を継承させることによって、プログラムの並列性，演算器の使用効率を向上させ、細粒度のスレッドに対しても高性能を並列処理を可能としたマルチスレッド実行方法を提供することにある。

【００２５】

【課題を解決するための手段】本発明の第１の発明は、論理的に複数のプログラムカウンタを有し、これらのプログラムカウンタに従った複数のスレッドの命令を同時にフェッチ，解釈，実行するスレッド実行手段（図２の５ａ，５ｂ）と、独立した論理的なレジスタファイル（図２の６ａ，６ｂ）とを備え、単一のスレッド中の指定された命令によって、新たなスレッド（子スレッド）の実行を開始する機能を備えたプロセッサにおいて、子スレッド実行開始命令を実行するスレッド（親スレッド）の命令実行時点のレジスタファイルの内容を子スレッドのレジスタファイルに継承することを特徴とする（図１）。

【００２６】また第２の発明は、第１の発明において、スレッド毎に複数のレジスタファイル（図４の１３ａ，１３ｂ）及びレジスタ選択手段（図４の１２ａ，１２ｂ）を備えると共に、レジスタ内容転送手段（図４の１５０）を備え、子スレッドは、前記レジスタ選択手段によって親スレッドのレジスタファイルを参照し、レジスタの変更が行われる毎に、前記レジスタ内容転送手段によって順次新スレッドのレジスタファイルに、変更する前のレジスタ内容を転送し、前記レジスタ選択手段の選択内容を子スレッド側のレジスタファイルに切替えることによってレジスタファイルを継承させることを特徴とする。

【００２７】また第３の発明は、第１の発明において、親スレッドのプログラムにおける子スレッド実行開始命令とその前後に配置される他の演算命令との間で、プログラムの意味を変えない範囲内において非プログラム順序に実行することを特徴とする（図８）。

【００２８】また第４の発明は、第３の発明において、演算の結果とレジスタ番号を指定するタグを一時的に格納するリオーダバッファを持ち、命令をデコードした際に、そのリオーダバッファのエントリを確保し、命令をプログラムで指定された順序ではなく、必要なレジスタの値が使用可能になったものからレジスタ及びリオーダバッファの当該命令よりも前のエントリから供給することによって演算を行い、演算結果を命令デコード時に確保したリオーダバッファのエントリに格納し、そのリオーダバッファからはプログラム順序でレジスタ内容の更新を行うことによって、プログラムの順序に従わずに処理を進める非プログラム順序実行型プロセッサにおいて、複数のプログラムカウンタを有し、これらのプログラムカウンタに従った複数のスレッドの命令を同時にフェッチ，解釈，実行するスレッド実行手段（図９の２１ａ，２１ｂ）と、複数のレジスタファイル（図９の２５ａ，２５ｂ）およびリオーダバッファ（図９の２４ａ，２４ｂ）を設け、親スレッドと子スレッドの両者のレジスタファイルおよびリオーダバッファからの内容出力を選択するレジスタデータセレクタ装置（２６ａ，２６ｂ）を設け、子スレッド生成命令が親スレッド内でプログラム順序で完了した時点で、親スレッドを実行しているレジスタファイルの内容を子スレッドを実行しているレジスタファイルにコピーすることによって、コピー前は親スレッドのレジスタファイル，リオーダバッファおよび子スレッドのリオーダバッファからレジスタ内容を選択し、コピー後は子スレッドのリオーダバッファおよびレジスタファイルからレジスタ内容を選択することによってレジスタの継承を行うことを特徴とする。

【００２９】また第５の発明は、第３の発明において、演算の結果とレジスタ番号を指定するタグを一時的に格納するリオーダバッファを持ち、命令をデコードした際に、そのリオーダバッファのエントリを確保し、命令をプログラムで指定された順序ではなく、必要なレジスタの値が使用可能になったものからレジスタ及びリオーダバッファの当該命令よりも前のエントリから供給することによって演算を行い、演算結果を命令デコード時に確保したリオーダバッファのエントリに格納し、そのリオーダバッファからはプログラム順序でレジスタ内容の更新を行うことによって、プログラムの順序に従わずに処理を進める非プログラム順序実行型プロセッサにおいて、複数のプログラムカウンタを有し、これらのプログラムカウンタに従った複数のスレッドの命令を同時にフェッチ，解釈，実行するスレッド実行手段（図１５の５０ａ，５０ｂ）と、複数のレジスタファイル（図１５の５４ａ，５４ｂ）およびリオーダバッファ（５３ａ，５３ｂ）を設け、親スレッドと子スレッドの両者のリオーダバッファ及び子スレッドのレジスタファイルからの内容出力を選択するレジスタデータセレクタ装置（５５ａ，５５ｂ）を設け、レジスタファイルの内容のコピーを子スレッド生成が行われた時点で行うこととして、その後は親スレッドのリオーダバッファから、親スレッドのレジスタファイルと子スレッドのレジスタファイルに書き込みを行い、スレッド生成命令がプログラム順序で終了する前は親スレッドのリオーダバッファ，子スレッドのレジスタファイルおよびリオーダバッファからレジスタ内容を選択し、プログラム順序で終了した後は子スレッドのリオーダバッファおよびレジスタファイルからレジスタ内容を選択することによってレジスタの継承を行うことを特徴とする。

【００３０】また第６の発明は、第４または第５の発明において、レジスタファイルのコピーを複数回の転送サイクルによって行うレジスタ内容転送手段（図１７の６０，６１等）を用い、レジスタ内容の転送が済んだレジスタファイル部分から、新スレッドにおいて、参照を許可することを特徴とする。

【００３１】また第７の発明は、第４，第５または第６の発明において、スレッド生成命令を実行した時点で、そのスレッドを実行できる資源が確保できない場合に、レジスタの内容を退避用レジスタファイル（図１９の７２）に蓄えることによって、プロセッサの許容数以上のスレッドが同時に存在し得るようにしたことを特徴とする。

【００３２】また第８の発明は、第３の発明において、プログラムから指定する論理レジスタとハードウェアに実行される物理レジスタとの対応関係を可変とし、この対応関係を記録，更新，検索するレジスタ写像テーブルを備え、論理レジスタに対して値の書き込みを行おうとする毎に物理レジスタとの新しい対応関係を生成し、対応する物理レジスタに値の書き込みが完了した時点で読み出しを許可し、命令がプログラム順序に完了した時点で、論理レジスタとの古い対応関係になっていた物理レジスタを未使用状態にすることによって、非プログラム順序で命令を実行するプロセッサにおいて、複数のプログラムカウンタを有し、これらのプログラムカウンタに従った複数のスレッドの命令を同時にファイル，解釈，実行するスレッド実行手段（図２０の７３ａ，７３ｂ）を設け、前記複数のスレッド実行手段から参照できる共有の物理レジスタファイル（図２０の７８）と、この共有の物理レジスタファイルの使用状態保持手段（図２０の８１，８２）と、複数のレジスタ写像テーブル（図２０の７６ａ，７６ｂ）とを設け、子スレッド実行開始命令を解釈した時点で、これらのレジスタ写像テーブル間で内容をコピーすることによってレジスタの継承を行い、継承した物理レジスタを親スレッド，子スレッド両者で対応する論理レジスタに書き込みを行った時点で、未使用状態にすることによってレジスタの継承を行うことを特徴とする。

【００３３】また第９の発明は、第８の発明において、レジスタ写像テーブルの内容のコピーを複数回の転送サ

イクルによって行い、新スレッドにおいて、対応関係の転送が済んだものから使用可能にすることを特徴とする。

【００３４】また第１０の発明は、第８または第９の発明において、スレッド生成命令を実行した時点で、そのスレッドを実行できる資源が確保できない場合に、レジスタ写像テーブルの内容を退避用レジスタ写像テーブル（図２４の９８）に蓄えることにより、プロセッサの許容数以上のスレッドが同時に存在し得るようにすることを特徴とする。

【００３５】また第１１の発明は、第２ないし第１０の発明において、親スレッドから子スレッドを生成する回数を高々１回に制限し、スレッドの消滅順序を親スレッドを子スレッドより先にすることによって、隣接するスレッド実行手段に限定してレジスタ内容継承を行うようにしたことを特徴とする。

【００３６】第１の発明ではフォーク命令を行った時点で、新規生成スレッドに対してレジスタ内容が継承される。この方法を用いる場合には、レジスタに格納できる範囲内のデータ数であれば、メモリに対するストア／ロードを省くことが可能になる。

【００３７】また第２の発明では、複数のレジスタファイルを選択する手段を用いて、レジスタファイルの内容の物理的なコピーを行わずに、レジスタ内容の継承を実現できる。

【００３８】また第３の発明では、プログラム順序において新規スレッドを生成する命令より前の命令の実行完了を待たずに、新規スレッドを生成する。また、新規スレッド生成命令より後続の命令がスレッド生成を行う前に実行完了できる。そして、このときに継承させるレジスタの内容はプログラム順序における新規スレッド実行命令時の内容とする処理方法を採用する。このような処理方法を実現するため、第４から第７の発明では、リオーダバッファを用いたｏｕｔ－ｏｆ－ｏｒｄｅｒ型実行を行い、レジスタファイルの内容をコピーするタイミングを工夫し、コピーが終了する前や、値が確定するまでは異なったレジスタファイルやリオーダバッファをアクセスするようにして、レジスタ内容を継承する。フォーク命令がプログラム順序に従ってレジスタ内容を更新した際に、レジスタの内容を確定させその後は新スレッド用のレジスタを用いることによってレジスタ内容を継承する。また、第８から第１０までの発明では、ソフトウェアによって指定する論理レジスタを物理レジスタに写像するためのレジスタ写像テーブルを用いたレジスタリネーミングを行い、このレジスタ写像テーブルを物理的に同時に実行するスレッド数に応じて複数用意する。新たなスレッドが生成される際には、この写像情報をコピーする。プロセッサに存在する物理的なレジスタはすべてスレッドから共有され、この写像情報によって各々のスレッドが使用する。

【００３９】以下に図面を参照して本発明をより具体的に説明するが、以下の開示は本発明の一実施例に過ぎず、本発明の技術的範囲をなんら限定するものではない。

【００４０】

【発明の実施の形態】図１は本発明の第１の発明におけるレジスタ内容の継承方法の概念図である。図１に示すように、新スレッドを生成するスレッド（親スレッド）１がその実行フローの実行途中においてスレッド生成命令（フォーク命令）２を実行して新スレッド（子スレッド）３を生成すると、子スレッド３のレジスタファイルに親スレッド１がフォーク命令２を実行した時点の内容を継承させる。

【００４１】図２は上記のようなレジスタ内容の継承方法を実現する２スレッド並列実行型プロセッサの実施例のブロック図である。この例のプロセッサは、スレッド管理手段４，スレッド実行手段５ａ，５ｂ，レジスタファイル６ａ，６ｂ，レジスタ内容一括転送手段７および共有メモリ８から構成される。

【００４２】スレッド管理手段４は、プロセッサ全体のスレッドの実行管理を行う。また、スレッド実行手段５ａ，５ｂ，レジスタファイル６ａ，６ｂは同時実行可能なスレッド数と同数用意され、通常のプロセッサ同様プログラムコード中の命令を実行し、レジスタファイル６ａ，６ｂおよび共有メモリ８に結果を書き込みながら処理を進める。

【００４３】以下、スレッド生成時の動作を時間順序を追って説明する。図３は図２に示した実施例のプロセッサにおけるスレッド生成時のタイミングチャートである。図３において、（Ａ）はプロセッサ全体のクロック、（Ｂ）はスレッド実行手段＃０　５ａの実行状態、（Ｃ）はレジスタファイル＃０　６ａのアクセス状態、（Ｄ）はスレッド実行手段＃１　５ｂの実行状況、（Ｅ）はレジスタファイル＃１６ｂのアクセス状態、（Ｆ）はスレッド管理手段４のアクセス状況を示す。

【００４４】図３のタイミングチャートの開始時点ではスレッド実行手段＃０　５ａにおいてのみ処理が行われており、スレッド実行手段＃１　５ｂは実行待ち状態である。従ってレジスタファイル＃１　６ｂに対するアクセスは生じない。

【００４５】サイクル６において、スレッド実行手段＃０　５ａはフォーク命令を実行すると、スレッド管理手段４に対してスレッド生成要求を行う。スレッド管理手段４は実行待ち状態のスレッド実行手段を検索し、新しいスレッドを実行するスレッド実行手段を決定する。図３の場合、（Ｄ）で示されるスレッド実行手段＃１５ｂが実行待ち状態であるので、ここで新しいスレッドを実行することを決定し、スレッド管理手段４はスレッド実行手段＃１　５ｂに対してスレッド起動要求を行う。同時に、スレッド実行手段４は、レジスタ内容一括転送手

段7に対してレジスタファイル#0　6aからレジスタファイル#1　6bへのレジスタ内容のコピー要求を行う。レジスタ内容一括転送手段7は、この要求に従いレジスタファイル#0　6aからレジスタファイル#1　6bへのレジスタ内容のコピーを行う。これらの動作によって、サイクル7からは、二つのスレッドが並列に実行され、子スレッド3では、初期値を共有メモリ8からロードすることによってレジスタファイル6にセットする必要が無くなり、親スレッド1では必要なデータをレジスタファイル6から共有メモリ8にストアする必要がなくなり、効率的な新スレッド生成が可能になる。なお、並列動作ではレジスタファイル6a，6bは独立に参照，更新されることになる。

【0046】次に、第2の発明について説明する。図4はその実施例の2スレッド並列実行型プロセッサのブロック図であり、スレッド実行手段を9a，9bの二つ持ち、各々のスレッド実行手段9a，9bは、命令キャッシュ10a，10b，命令デコーダ11a，11b，レジスタ選択手段12a，12b，レジスタファイル13a，13b，演算ユニット14a，14bを各々持っている。また、共有のスレッド管理手段15とレジスタ内容転送手段150が存在する。なお、共有メモリについては直接関係しないので、図4からは省いてある。

【0047】図5はレジスタ選択手段#1　12bの構成例を示すブロック図である。レジスタ選択手段#0　12aも同等の構造であり、レジスタ選択テーブル16に各レジスタの選択ビット17を持っている。本実施例の場合、レジスタ選択ビット17が0の場合にはレジスタファイル#0　13aを、1の場合にはレジスタファイル#1　13bを選択する。

【0048】図6はレジスタファイル#0　13aの構成例を示すブロック図であり、レジスタファイル#1　13bも同等の構造である。レジスタファイル#0　13aは、各レジスタ毎にレジスタ内容メモリ18の他に、転送終了ビット19，更新ビット20を持つ。転送終了ビット19は、スレッド管理手段15によって、子スレッド生成時に初期化（リセット）され、レジスタ内容メモリ18の内容を他のレジスタファイル13に転送した際にセットされる。また、更新ビット20は、スレッド起動時に初期化（リセット）され、レジスタを自分の属する演算ユニット14で更新した場合にセットされる。

【0049】以下、スレッド生成時のレジスタ内容の継承方法を具体例を用いて説明する。図7は、図4に示した実施例のプロセッサのタイミングチャートである。図7において、（A）はプロセッサ全体のクロック、（B）は演算ユニット#0　14a（親スレッド）で実行している命令もしくは書き込みされるレジスタ番号、（C）は親スレッドによるレジスタファイル#0　13aに対するレジスタ内容の書き込み、（D）はレジスタ内容の転送によるレジスタファイル#0　13aに対するレジスタ内容の読み出し、（E）はレジスタファイル#0　13a中の転送終了ビット19の状態、（F）はスレッド管理手段15のアクセス状況、（G）は演算ユニット#1　14b（子スレッド）で実行している命令もしくは読み出し／書き込みされるレジスタ番号、（H），（I）は子スレッドによるレジスタファイル13の読み出し、（J）はレジスタファイル#1　13bへの書き込み、（K）はレジスタファイル#1　13b中の更新ビット20の状態、（L）はレジスタ選択手段#1　12b中のレジスタ選択ビット17の状態を示す。

【0050】図7のタイミングチャートの開始時点では、スレッドはスレッド実行手段#09aのみで実行されている。このスレッドを親スレッドとしている。サイクル2において、親スレッドはフォーク命令を実行する。スレッド管理手段15は、レジスタファイル#0　13aの転送終了ビット19と、レジスタファイル#1　13bの更新ビット20と、レジスタ選択手段#1　12b中のレジスタ選択ビット17の初期化要求を行う。このフォーク命令の実行によってサイクル3以降はサイクル2のフォーク命令による子スレッドがスレッド実行手段#1　9bで実行が開始される。

【0051】サイクル3において親スレッドを実行しているスレッド実行手段#0　9aではレジスタr1に対する書き込みが生じる前に、スレッド実行手段#1　9bにr1のレジスタ内容を転送するために読み出しが行われる。これがサイクル3前半の（D）に示されている。この内容はスレッド実行手段#1　9bのレジスタファイル#1　13bに転送され（J）で示されるように書き込みが行われる。同時に、レジスタファイル#0　13aの転送終了ビット19と、レジスタファイル#1　13bの更新ビット20のレジスタr1のエントリをセットし、レジスタ選択手段#1　12b中のレジスタ選択ビット17をr1またはレジスタファイル#1　13bから選択するようにセットする。

【0052】また、同サイクルにおいて子スレッドを実行しているスレッド実行手段#19bではレジスタr7，r10を読み出し参照する。この読み出しはレジスタ選択手段#1　12bの内容に従ってレジスタファイル#0　13aから読み出す。また、同サイクルにおける子スレッドはr3に対して書き込みを行うが、この際にレジスタ選択手段#1　12b中のレジスタ選択ビット17のr3をレジスタファイル#1　13bから選択するようにセットする。同時にレジスタファイル#1　13bの更新ビット20のr3エントリをセットする。

【0053】サイクル4も同様の処理が行われ、r5の内容が転送され、レジスタファイル#0　13aの転送終了ビット19と、レジスタファイル#1　13bの更新ビット20と、レジスタ選択手段#1　12b中のレ

15

ジスタ選択ビット１７のレジスタｒ５のエントリがセットされる。また、同サイクルにおいてレジスタ選択手段＃１　１２ｂ中のレジスタ選択ビット１７と、レジスタファイル＃１　１３ｂの更新ビット２０のｒ２エントリをセットする。

【００５４】次に、サイクル５では親スレッドを実行しているスレッド実行手段＃０　９ａでは再びレジスタｒ１に対する書き込みを行う命令を実行する。しかしながら、レジスタファイル＃０　１３ａの転送終了ビット１９のｒ１エントリは既にセットされているので、レジスタ内容の転送は行われない。また、同サイクルにおいて子スレッドを実行しているスレッド実行手段＃１　９ｂではレジスタｒ３，ｒ５を読み出し参照する。この読み出しはレジスタ選択手段＃１　１２ｂの該当エントリはセットされているのでレジスタファイル＃１　１３ｂから読み出す。

【００５５】次に、サイクル６では親スレッドを実行しているスレッド実行手段＃０　９ａではレジスタｒ２に対する書き込みを行う命令を実行し、レジスタの内容が転送される。しかしながら、レジスタファイル＃１　１３ｂの更新ビット２０のｒ２エントリは既にセットされているので、レジスタファイル＃１　１３ｂのレジスタ内容メモリ１８には書き込みは生じない。

【００５６】以上の処理によって、スレッド生成のフォーク命令実行時にレジスタファイルの全内容を一括転送する必要がなくなり、レジスタファイル間の転送バンド幅を削減しつつ、レジスタ内容を子スレッドに継承することが可能となる。

【００５７】本実施例は２スレッド並列実行型プロセッサについて説明したが、３スレッド以上の並列処理を行う際には、スレッド管理ユニットによってレジスタの転送先を制御し、レジスタファイル間はバスなどで接続し多対多転送をサポートするような拡張を施せば良い。また、レジスタ選択手段ではどのレジスタファイルを選択するかを示すレジスタ選択ビットが複数ビットに格納されるが、本質的にレジスタ選択手段を用いることには相違なく、本発明の範囲内である。

【００５８】次に第３の発明について説明する。第３の発明は、ソフトウェアによるプログラミングモデルでは第１の発明の方法に従うが、ハードウェアによる実際の処理は図８に示すように必ずしもプログラム順序に実行を行わないハードウェア処理方法である。つまり、子スレッドには図８で示されたパイプラインの実行イメージ図において、０ｘ００番地，０ｘ０４番地の命令実行によるレジスタ内容を継承させ、０ｘ０ｃ番地，０ｘ１０番地の命令実行によるレジスタ内容は継承しない処理方法である。第４の発明以降はこの処理方法に基づく実施形態についての発明である。

【００５９】次に第４の発明について説明する。図９はその実施例のブロック図であり、例として２スレッド並列実行型プロセッサを採り上げている。図９のプロセッサは、スレッド実行手段を＃０　２１ａと＃１　21b の二つ持ち、各々は命令キャッシュ２２ａ，２２ｂ，命令デコーダ２３ａ，２３ｂ，リオーダバッファ２４ａ，２４ｂ，レジスタファイル２５ａ，２５ｂ，レジスタデータセレクタ装置２６ａ，２６ｂ，命令キュー２７ａ，２７ｂ，演算ユニット２８ａ，２８ｂを持っている。また、共有のスレッド管理手段２９が存在する。

【００６０】図１０は、命令キュー＃０　２７ａの構成例を示すブロック図であり、命令キュー＃１　２７ｂも同様の構造である。命令キュー２７は、デコードした命令について、必要なレジスタの値が確定し演算できるようになるまで待ち合わせる機構であり、キューエントリ格納論理３０，発行命令決定論理３１，命令キューエントリ３２から構成される。

【００６１】命令キューエントリ３２は、命令発行に必要な情報を蓄えておくもので、エントリ有効ビット３３，レジスタ内容／リオーダバッファタグ格納メモリ３４，レジスタ内容有効ビット３５，結果書き込み場所指定タグ３６，命令コード格納メモリ３７のエントリから構成される。エントリ有効ビット３３はそのエントリに格納されているデータが有効かどうかを示す。レジスタ内容／リオーダバッファタグ格納メモリ３４は、レジスタ内容有効ビット３５の状態によって格納されている値が異なる。もし、セットされている場合には対応する命令の演算に必要なデータ、セットされていない場合にはリオーダバッファ２４が付加したタグ番号である。また、結果書き込み場所指定タグ３６はこの命令の実行による結果を格納するリオーダバッファ２４のタグが格納され、命令コード格納メモリ３７のエントリは、命令デコーダ２３でデコードされた命令種類のコードが格納される。

【００６２】キューエントリ格納論理３０は、空いているエントリに対して必要なデータを格納する論理であり、エントリ有効ビット３３の無効なエントリのうち一つを決定し、レジスタデータセレクタ装置２６，命令デコーダ２３からの情報を格納する。発行命令決定論理３１は、レジスタ内容有効ビット３５が有効になったエントリの中から、発行する命令を決定し、演算ユニット２８に対してその命令を発行する。

【００６３】また命令キューエントリ３２は、連想メモリ形態となっており、レジスタ内容有効ビット３５がセットされていないエントリに対して、演算ユニット２８から送られてくるタグ番号と自分の持つタグ番号を比較して、同一の場合には、演算結果をレジスタ内容／リオーダバッファタグ格納メモリ３４に格納し、レジスタ内容有効ビット３５をセットする。

【００６４】図１１は、リオーダバッファ＃０　２４ａの構成例を示すブロック図であり、リオーダバッファ＃１　２４ｂも同様の構造である。リオーダバッファ２４

は、プログラム順序に従わずに先行して確定したレジスタ値を保持しておき、プログラム順序に従ってレジスタファイル２５に書き戻す機構であり、レジスタデータ供給決定論理３８，エントリシフト制御論理３９，リオーダバッファエントリ４０から構成される。

【００６５】リオーダバッファエントリ４０は、そのエントリに格納されたデータが有効かどうかを示すエントリ有効ビット４１，演算ユニット２８からの結果の書き戻しの際に用いる結果書き込み指定タグ４２，レジスタファイル２５への書き戻し場所を指定するための、レジスタ番号４３，フォーク命令後でレジスタデータを子スレッドに継承させる必要を示すスレッド生成ビット４４，演算ユニット２８からの結果を格納する演算データ格納メモリ４６と格納されるとセットされる演算データ格納メモリ有効ビット４５のエントリから構成される。

【００６６】リオーダバッファ２４は、命令デコーダ２３から命令を受け取ると、プログラムの命令順序に従ってエントリが確保される。つまりエントリはプログラム命令順に整列していることになる。また、エントリが確保できない場合には、命令デコードが停止する。また、フォーク命令実行後の命令を格納する際には、スレッド生成ビット４４をセットする。

【００６７】また、命令デコーダ２３からソースレジスタの参照番号も同時に受け取り、これとリオーダバッファエントリ４０中のレジスタ番号４３を比較する。同じエントリが存在すれば、その中で最も新しいエントリの演算データ格納メモリ４６か結果書き込み指定タグ４２の内容をレジスタデータセレクタ装置２６に送る。演算データ格納メモリ４６の内容が送られる場合は、該当エントリの演算が終了しており、演算データ格納メモリ有効ビット４５がセットされている場合である。また、他のスレッド実行手段２１のレジスタデータセレクタ２６に送る場合には、スレッド生成ビット４４がセットされていないエントリから選択する。これによって、フォーク命令後の親スレッドのレジスタ変更データが子スレッドに送られることを防ぐ。これらの処理をレジスタデータ供給決定論理３８が行う。

【００６８】演算結果が演算ユニット２８から来た場合には、結果書き込み指定タグ４２の一致するエントリの演算データ格納メモリ４６に書き込み、演算データ格納メモリ有効ビット４５をセットする。また、演算が終わったものをプログラム順序に従ってレジスタファイル２５に書き戻し、エントリシフト制御論理３９によって、先頭からエントリをシフトする。なお、シフトを行わずにリング状のバッファ管理を行うことも可能である。また、フォーク命令がプログラム順序で完全に終了した場合には、すべてのエントリのスレッド生成ビット４４をリセットする。

【００６９】レジスタデータセレクタ装置＃１　２６ｂは、図１２に示す論理によって、命令キュー＃１　２７ｂに供給するデータをリオーダバッファ＃１　２４ｂ，レジスタファイル＃１　２５ｂ，リオーダバッファ＃０，２４ａ，レジスタファイル＃０　２５ａの入力データから選択する。レジスタデータ選択装置＃０　２６ａも同様の論理である。

【００７０】以下、図９のプロセッサの実際の動作について説明する。図９のプロセッサはスレッド生成時以外の通常時はスレッド実行手段２１ａ，２１ｂでそれぞれ独立して処理を進める。従って、レジスタデータセレクタ装置２６には、他方のスレッド実行手段２１のリオーダバッファ２４，レジスタファイル２５からデータが供給されることはなく、自らのスレッド実行手段２１のリオーダバッファ２４，レジスタファイル２５からのデータを選択して、レジスタデータが使用可能になった命令から実行が行われる。この点においては、リオーダバッファを用いてｏｕｔ－ｏｆ－ｏｒｄｅｒ実行を行う従来のスーパスカラプロセッサの処理方法と相違はない。

【００７１】フォーク時の動作モデルを図１３に示す。フォーク命令がスレッド実行手段＃０　２１ａでデコードされると、スレッド管理手段２９によってスレッド実行手段＃１　２１ｂで子スレッドの実行が開始される。しかしながら、レジスタファイル＃１　２５ｂには、親レジスタのデータが継承されていないので、レジスタファイル＃０　２５ａとリオーダバッファ＃０　２４ａのデータをレジスタデータセレクタ装置＃１　２６ｂで選択して用いて演算を行うことになる。この演算の結果はリオーダバッファ＃１　２４ｂに格納される。従って、後続命令は、レジスタファイル＃０　２５ａとリオーダバッファ＃０　２４ａとリオーダバッファ＃１　２４ｂのデータをレジスタデータセレクタ装置＃１　２６ｂで選択して用いて演算を行う。

【００７２】一方、親スレッド側でフォーク命令がプログラム順序で完全終了した時点で、レジスタファイル＃０　２５ｂのすべての値がフォーク命令実行時の値と確定する。この時に、スレッド管理手段２９によってレジスタファイル＃０　２５ａの内容をレジスタファイル＃１　２５ｂにコピーする。この動作が終了すると、レジスタファイル＃１　２５ｂには親レジスタの値が継承されたことになるので、レジスタファイル＃１　２５ｂとリオーダバッファ＃１　２４ｂのデータをレジスタデータセレクタ装置＃１　２６ｂで選択して用いて演算を行うことになる。また、この時点で、リオーダバッファ＃１　２４ｂのレジスタファイル＃１　２５ｂへのレジスタアップデート動作が可能になる。これによって、ｏｕｔ－ｏｆ－ｏｒｄｅｒ実行を行うプロセッサにおいてもレジスタ内容の継承が可能になる。

【００７３】以上が図９に示した２スレッド並列実行型プロセッサの実施例の説明である。

【００７４】本実施例の他の構成として、命令キュー２７，演算ユニット２８をスレッド間で共有する手法があ

げられる。図１４は４スレッド並列実行２演算ユニット型プロセッサの場合の実施例のブロック図である。図１４では、命令キャッシュ，命令デコーダ，スレッド管理手段等は省略しているが、この部分の実際の構成は図９と同様である。本実施例では新たに、リオーダバッファ２４への結果振り分け装置４７ａ，４７ｂを用意し、演算ユニット２８からの結果を振り分けている。このため、命令キュー４９には、どの命令デコーダによってデコードされた命令であるかを記憶するエントリの追加が必要である。また、レジスタデータ選択装置４８ａ，４８ｂは、選択するレジスタ値がどの命令デコーダの命令について取り扱っているかの情報が必要となる。また、４スレッド構成としたので、レジスタファイル２５の転送手段が多対多転送を実現するような構成にする必要があり、具体的にはバスなどを用いる必要がある。

【００７５】次に第５の発明について説明する。図１５は、その実施例のブロック図であり、例として２スレッド並列実行型プロセッサを採り上げている。主たる構成は、図９と同様である。構成上の相違点は、リオーダバッファ５３から他のスレッド実行手段５０のレジスタファイル５４に対してレジスタ内容更新用のパスが存在すること、及びレジスタデータセレクタ装置５５への入力に他のスレッド実行手段５０のレジスタファイル５４からの入力が存在しないことである。

【００７６】フォーク時の動作モデルを図１６に示す。親スレッドでフォーク命令がスレッド実行手段＃０　５０ａでデコードされると、スレッド管理手段５８によってスレッド実行手段＃１　５０ｂで子スレッドの実行が開始され、同時にスレッド管理手段５８によってレジスタファイル＃０　５４ａの内容をレジスタファイル＃１　５４ｂにコピーする。

【００７７】しかしながら、レジスタファイル＃０　５４ａでは、フォーク命令よりも前の命令によるレジスタ更新が完全に行われておらず、更新データが親スレッドのスレッド実行手段５０のリオーダバッファ＃０　５３ａに存在するか、まだ演算が行われていない可能性がある。従って、レジスタデータセレクタ装置＃１　５５ｂでは、子スレッドのレジスタファイル＃１　５４ｂとリオーダバッファ＃０５３ａとリオーダバッファ＃１　５３ｂのデータを選択して用いて演算を行う。

【００７８】一方、親スレッド側でフォーク命令がプログラム順序で完全終了した時点で、リオーダバッファ＃０　５３ａのデータをレジスタファイル＃１　５４ｂに供給する必要はなくなるので、レジスタファイル＃１　５４ｂとリオーダバッファ＃１　５３ｂのデータをレジスタデータセレクタ装置＃１　５５ｂで選択して用いて演算を行うことになる。また、この時点で、リオーダバッファ＃１　５３ｂのレジスタファイル＃１　５４ｂへのレジスタアップデート動作が可能になりｏｕｔ－ｏｆ－ｏｒｄｅｒ実行を行うプロセッサにおいてもレジスタ

内容の継承が実現される。

【００７９】図９，図１５に示したプロセッサでは、レジスタファイルの内容のコピーをフォーク命令の完了時／デコード時に一度行うことを前提にしたレジスタファイル間の転送手段を前提にしていた。しかしながら、現実にはレジスタファイルの内容のすべてを同時にコピーするには、高バンド幅の転送手段が必要となる。

【００８０】第６の発明は、このような問題に対して、レジスタファイルの転送を複数回に分割することによって対処するものである。図１７は、第６の発明の実施例のブロック図である。同図は、１６ワードレジスタファイル，４ワード４回転送型レジスタ転送手段の構成例を示しおり、送信側のレジスタファイル＃０　５９ａ，受信側のレジスタファイル＃１　５９ｂ，マルチプレクサ６０，デバイダ６１，参照許可ビット６２ａ，６２ｂから構成される。

【００８１】図１８は、図１７の実施例のタイミングチャートである。図１８において、（Ａ）から（Ｄ）は転送中のレジスタファイルを示し、（Ｅ）はマルチプレクサ６０からデバイダ６１へ現在どのレジスタを転送しているかを伝達するための情報線の内容を示し、（Ｆ）から（Ｉ）はレジスタファイル＃１　５９ｂの参照許可ビット６２ｂの状態を示している。

【００８２】レジスタファイル５９のコピーが開始されると、レジスタファイルはｒ０－ｒ３，ｒ４－ｒ７，ｒ８－ｒ１１，ｒ１２－ｒ１５の４回に分割して転送される。転送中は送信側のレジスタファイル＃０　５９ａの更新と、レジスタファイル＃１　５９ｂの参照は禁止される。転送が終了次第順次レジスタファイル＃０　５９ａの更新とレジスタファイル＃１　５９ｂの参照が許可される。この参照の許可は参照許可ビット６２ａ，６２ｂをセットすることによって行われる。

【００８３】これによって、レジスタ転送のバンド幅を低減しながらレジスタ転送が可能になる。なお、レジスタ転送時にレジスタファイルのアクセスが禁止されるサイクルが増加するが、順次レジスタアクセス禁止が解除されるので、その特性に合わせたコードスケジュールを行い、使用できないレジスタに対するアクセスを遅らせることにより、このレイテンシを或る程度隠蔽することが可能になる。

【００８４】次に、第７の発明について説明する。図１９はその実施例のブロック図である。基本的には、第４の発明にかかる図９の実施例とほぼ同様であるが、退避用レジスタファイル７２が新たに設けられている。

【００８５】空いているスレッド実行手段６３が存在しない時にさらにフォーク命令が実行された場合、フォーク命令を実行したスレッド実行手段６３がレジスタファイル６７の確定した時にこの退避用レジスタファイル７２にコピーを行う。このことによって、システム管理ソフトウェアの介入なく、スレッド実行手段６３の数を超

えるスレッド生成要求に対処する。スレッド実行手段６３に空きができた時点で、退避用レジスタファイル７２から、空いたスレッド実行手段６３のレジスタファイル６７に対してコピーを行い、退避していたスレッドの実行を再開する。これらの管理はスレッド管理手段７１によって行われる。

【００８６】次に、第８の発明について説明する。図２０は、その実施例の一例を示す２スレッド並列実行型プロセッサのブロック図である。図２０のプロセッサはスレッド実行手段７３ａ，７３ｂと共有の物理レジスタファイル７８，レジスタビジーテーブル８１，レジスタフリーテーブル８２，スレッド管理手段８３から構成される。

【００８７】スレッド実行手段７３ａ，７３ｂはそれぞれ、命令キャッシュ７４ａ，７４ｂ，命令デコーダ７５ａ，７５ｂ，レジスタ写像テーブル７６ａ，７６ｂ，命令キュー７７ａ，７７ｂ，演算ユニット７９ａ，７９ｂ，有効命令順序バッファ８ａ，８０ｂから構成される。

【００８８】この実施例では、レジスタをソフトウェアからアクセスする論理レジスタと、ハードウェア的にレジスタ内容を保持する物理レジスタを分離し、その写像関係をレジスタ写像テーブル７６に保持する。図２１は、レジスタ写像テーブル７６ａの詳細なブロック図である。レジスタ写像テーブル７６ｂも同様の構造である。レジスタ写像テーブル７６は、論理レジスタ数分の物理レジスタ番号エントリを持っており、論理レジスタ番号を物理レジスタ番号に変換する。

【００８９】どのスレッド実行手段７３が、どのレジスタを使用しているかという情報を管理するのが、レジスタフリーテーブル８２である。図２２は、レジスタフリーテーブル８２の詳細なブロック図である。レジスタフリーテーブル８２は、レジスタフリー決定論理８４，フリーレジスタ検索論理８５，状態テーブル８６から構成される。

【００９０】命令が命令デコーダ７５によってデコードされると、読み出し参照する論理レジスタ番号と書き込み参照する論理レジスタ番号が確定する。読み出し参照を行う論理レジスタ番号は、直ちにレジスタ写像テーブル７６によって物理レジスタ番号に変換され、命令キューに格納される。

【００９１】書き込み参照する論理レジスタに対しては、新しい物理レジスタを確保して用いる。これは、ｏｕｔ－ｏｆ－ｏｒｄｅｒ実行を行った際に、現在デコードした命令により前の命令が同じ論理レジスタを参照する際の正当性を維持するためである。例えば、

１０　　ａｄｄ　　ｒ１←ｒ２＋ｒ３
１４　　ｓｕｂ　　ｒ３←ｒ４－ｒ２

という命令をｏｕｔ－ｏｆ－ｏｒｄｅｒで実行する場合には、１４番地の命令を実行して、更新されたｒ３の値

を１０番地の命令が読み出すとプログラムの正当性が維持できなくなる。そこで、１０番地のｒ３と１４番地のｒ３を異なった物理レジスタに写像することによって、１４番地の命令を実行しても実行前のｒ３の値を保持しておき、１０番地の実行時には前のレジスタ写像関係から前のｒ３の値を読み出し参照するということを行う。このため、現在未使用の物理レジスタをレジスタフリーテーブル８２から確保し、レジスタ写像テーブル７６の写像情報を新しい論理レジスター物理レジスタ対応関係に更新する。

【００９２】命令デコード時には、有効命令順序バッファ８０に対しても命令のプログラム順序に従って必要な情報を記憶させる。ここでは、デコード命令によって新しい論理レジスター物理レジスタ写像関係が生成された場合、その命令実行前のレジスタ写像関係を併せて記憶させる。これは、物理レジスタの開放や処理の取消時に必要になるためである。

【００９３】新たに確保した物理レジスタは実際に書き込みが生じるまでは読み出し参照を禁止する。この書き込みが生じたか否かをレジスタビジーテーブル８１で管理する。確保した物理レジスタは最初ビジー状態で読み出し参照を禁止する。その後、書き込みが生じた後にフリー状態に変更して読み出し参照を許可する。図２３はレジスタビジーテーブル８１の詳細なブロック図である。レジスタビジーテーブル８１の管理するレジスタ数は物理レジスタ本数に対応する。

【００９４】物理レジスタはプログラム順序で命令を完全に終了させる際に開放を行う。例えば、

１０　　ａｄｄ　　ｒ１←ｒ２＋ｒ３
１４　　ｓｕｂ　　ｒ３←ｒ４－ｒ２

の例の場合、１４番地の命令が終了時に１０番地までに用いていたｒ３に対応する物理レジスタを開放する。１４番地で確保したｒ３の物理レジスタを開放するのは、その後のｒ３を更新する命令がプログラム順序で完全終了する際である。但し、この開放は子スレッドにレジスタ内容が継承されていない場合である。子スレッドに継承されている場合、親スレッド，子スレッド両者が開放された時に完全に開放されたことになる。つまり、状態テーブル８６のすべてのエントリがリセットされた場合に、当該レジスタは空き状態になる。

【００９５】以下、図２０のプロセッサの実際の動作について説明する。図２０のプロセッサは、スレッド生成と終了時以外はスレッド実行手段７３で独立して処理を進める。物理レジスタファイル７８，レジスタビジーテーブル８１，レジスタフリーテーブル８２は共有であるが、スレッド毎に異なった要求を同時に処理するだけである。従って、この点においては、レジスタ写像テーブル７６を用いてレジスタリネーミングを行う従来のスーパスカラプロセッサの処理方法と何ら相違はない。

【００９６】さて、図２０のプロセッサでは、フォーク

命令をデコードすると、レジスタ写像テーブル７６の内容を他の空いているスレッド実行手段７３のレジスタ写像テーブル７６にコピーする。同時にレジスタフリーテーブル８２に対してもコピーを行う。この情報を受けたレジスタフリーテーブル８２は、子スレッドのスレッド実行手段における状態テーブルをレジスタ使用中にセットする。レジスタ写像テーブル７６はデコード時にプログラム順序に従って内容が変更されるので、フォーク命令デコード時に、正しい写像情報を保持している。従って、子スレッドからも親スレッド同様に通常の物理レジスタファイル７８の参照が可能である。また、スレッド生成後は同一論理レジスタへの書き込みを行った物理レジスタへマッピングするので、各々のスレッドで独立した処理が可能である。スレッド終了時には、レジスタフリーテーブル８２の当該スレッド実行手段７３の状態テーブル８６をクリアする。

【００９７】このように本実施例では写像関係をコピーすることによって、ｏｕｔ－ｏｆ－ｏｒｄｅｒにおけるレジスタ内容の継承を実現する。本実施例では、第４から第７までの発明に比して、コピーする情報量が少ない。また、親スレッド側でフォーク命令がプログラム順序で終了するまでの特別な制御が不要であるという特徴がある。

【００９８】次に、第９の発明について説明する。この発明は第６の発明とほぼ同様の思想であるが、第８の発明に準じ、レジスタファイルの内容転送の代わりにレジスタ写像テーブルの内容を複数回に分割して実現するものである。

【００９９】これによって、レジスタ転送のバンド幅を一層低減しながらレジスタ内容の継承が可能になる。なお、レジスタ写像テーブル転送時にレジスタファイルのアクセスが禁止されるサイクルが増加するが、順次レジスタアクセス禁止が解除されるので、その特性に合わせたコードスケジューリングを行い、使用できないレジスタに対するアクセスを遅らせることにより、第６の発明同様、このレイテンシをある程度隠蔽することが可能になる。

【０１００】次に、第１０の発明について説明する。図２４は、この発明の実施例を示す２スレッド並列実行型プロセッサのブロック図である。本発明は、基本的には第８の発明にかかる図２０の実施例とほぼ同様であるが、退避用レジスタ写像テーブル９８を新たに設け、レジスタフリーテーブル９６に対して、退避用レジスタ状態テーブルを付加する。

【０１０１】空いているスレッド実行手段８７が存在しない時にさらにフォーク命令が実行された場合、フォーク命令を実行したプロセッサのフォーク命令デコード時にレジスタ写像テーブル９０の内容を退避用レジスタ写像テーブル９８にコピーする。このことによって、システム管理ソフトウェアの介入なく、スレッド実行手段８７の数を超えるスレッド生成要求に対処する。スレッド実行手段８７に空きができた時点で、退避用レジスタ写像テーブル９８から、空いたスレッド実行手段８７のレジスタ写像テーブル９０に対してコピーを行い、退避していたスレッドの実行を再開する。これらの管理はスレッド管理手段９７によって行われる。

【０１０２】次に第１１の発明について説明する。第１１の発明は、１スレッドでフォーク命令によって子スレッドを生成する回数を１回にし、さらに子スレッドは親スレッドが終了するまで終了できないとすることによって得られる効果についての発明である。子スレッドの生成を１回にすることにより、スレッドの生成／消滅は逐次的に行われるようになる。このことを４スレッド並列実行環境に対して適用したモデルが図２５である。

【０１０３】図２５に示すように、フォークによって生成される子スレッドの生成先は隣接するスレッド実行手段に限定できる。すなわち、スレッド実行手段＃０からは必ずスレッド実行手段＃１に対してフォークを行い、スレッド実行手段＃１からはスレッド実行手段＃２へ、スレッド実行手段＃２からはスレッド実行手段＃３へ、スレッド実行手段＃３からはスレッド実行手段＃０へというようにスレッドはリング状にスレッド実行手段に展開される。従って、第４から第１０の発明におけるレジスタファイルもしくはレジスタ写像テーブルの内容転送手段を多対多構造からリング状の転送手段に簡単化することが可能となる。

【０１０４】図２６は図１４の実施例に図２５の技術を適用した場合のレジスタファイル回りのブロック図である。図２６においてレジスタファイル９９は、リング状レジスタファイル転送手段１０３によって結合している。従って、物理的なレジスタファイル９９の位置を工夫することによって、より効率的なハードウェア実装が可能となる。

【０１０５】次に、本発明のその他の実施の形態について説明する。一つの他の実施の形態としては、第８の発明に対して、第１１の発明で用いた子スレッド生成１回限定の特徴を追加し、フォーク時に物理レジスタ継承情報をレジスタフリーテーブルに伝達することなく、論理レジスター物理レジスタのリネーミングを実現しつつ、子スレッドへのレジスタ内容継承を実現するものがある。図２７にこの実施の形態の実施例として２スレッド並列実行型のプロセッサの構成例を、図２８にレジスタフリーテーブル１１３の詳細図を示す。図２７のプロセッサは、基本的には図２０のプロセッサと同一構造であるが、レジスタ写像テーブル１０７からレジスタフリーテーブル１１３へ、フォーク時に物理レジスタの継承を伝える手段が省略されている。

【０１０６】また、レジスタフリーテーブル１１３は、図２８に示すように、状態テーブル１１７の各スレッド実行手段１０４のエントリが２ビットに拡張されてい

る。このビットの意味を表１に示す。

【０１０７】

【表１】

ビット	オーナー権	意味
00	×	当該スレッド実行手段はこの物理レジスタを確保していない
01	○	当該スレッド実行手段はこの物理レジスタをフォーク後に確保した
10	×	当該スレッド実行手段はこの物理レジスタをオーナー権なしでフリーした
11	○	当該スレッド実行手段はこの物理レジスタをフォーク前に確保した

【０１０８】レジスタ確保時には、既にフォークを行ったスレッドかフォークを行っていないスレッドかで状態テーブル１１７にセットされる値は異なる。従って、フォーク前か後かの情報をスレッド管理手段１１４から得る必要がある。また、表１中のオーナ権はレジスタを確保したスレッド実行手段１０４の状態テーブル１１７に付与されるものであるが、その後のレジスタ継承によって、子スレッド以下に委譲される場合もある。各スレッド実行手段１０４の状態テーブル１１７がすべて００の場合、対応するエントリの物理レジスタは未使用状態である。

【０１０９】以下、レジスタ開放論理について説明する。この論理はレジスタフリー決定論理１１５によって決定する。この論理の説明では、Ｎスレッド同時実行プロセッサモデルとしている。これは、２スレッド同時実行モデルでは、論理が簡単化されてしまうためである。またスレッドは、親スレッド→子スレッド→孫スレッドという順に生成されたものとする。ここで、子スレッドにおいて命令プログラム順序終了時のレジスタ開放時の論理を表２に示す。

【０１１０】

【表２】

レジスタ確保時	状態遷移前 親	子	孫	レジスタ開放時	状態遷移後 親	子	孫	コメント
フォーク前	00	11	00	フォーク前	00	00	00	レジスタ完全開放
フォーク後	00	01	00	フォーク後	00	00	00	レジスタ完全開放
フォーク前	00	11	00	フォーク後	00	00	11	オーナー権孫スレッドへ委譲
フォーク前	00	11	10	フォーク後 孫開放後	00	00	00	レジスタ完全開放
親確保	11	00	00	フォーク前	11	10	00	子スレッド先行開放
親確保	11	00	00	フォーク後 孫開放前	11	00	11	子スレッド先行開放／孫２重オーナ権
親確保	11	00	10	フォーク後 孫開放後	00	00	00	レジスタ完全開放

【０１１１】この表２から明らかなように、孫スレッドへのレジスタ継承が行われている際には、親フィールド，子フィールド，孫フィールドの３種類のフィールド値と開放しようとしているスレッドのフォーク状態によって状態遷移が決定される。これ以外の状態遷移は通常ではあり得ずエラーとなる。

【０１１２】この論理を用いることによって、フォーク命令実行時にレジスタフリーテーブル１１３は多数の状態テーブル１１７の内容を変更することなく、レジスタの継承／開放が可能になる。

【０１１３】さらに他の実施の形態では、図２９に示したように、レジスタ使用中ビット１２１を付加する。このレジスタ使用中ビットは、物理レジスタを確保する際に、フリーレジスタ検索論理１１９によって未使用のものを見つけ、使用中にセットする。このことによって状態テーブル１２０のすべてのエントリが００もしくは未使用であるという状態のレジスタを検索する論理が簡単化される。レジスタ使用中ビット１２１を使用中状態から、未使用状態にセットするのは、レジスタフリー決定論理１１８が、状態テーブル１２０の内容を書き換える際に同時に行う。

【０１１４】

【発明の効果】以上説明したように、本発明によれば、スレッドを並列に処理する際に、親スレッドから子スレッドに対して共有メモリを介さずにレジスタ内容の継承が可能になり、スレッド生成時のオーバヘッドを減らすことができる。また、このレジスタ内容の継承をｏｕｔ－ｏｆ－ｏｒｄｅｒ実行を行うプロセッサに対しても、フォーク命令前後間においても実現したため、スレッド生成にともなうオーバヘッドを減らすことが可能になり、粒度の細かいスレッドに対してもスレッドレベル並列処理による処理速度の向上が実現できる。

【図面の簡単な説明】

【図１】第１の発明におけるレジスタ内容の継承方法の概念図である。

【図２】第１の発明におけるレジスタ内容の継承方法を

実現する２スレッド並列実行型プロセッサの実施例のブロック図である。

【図３】図２に示した実施例のプロセッサにおけるスレッド生成時のタイミングチャートである。

【図４】第２の発明を適用した２スレッド並列実行型プロセッサの実施例のブロック図である。

【図５】レジスタ選択手段＃１　１２ｂの構成例を示すブロック図である。

【図６】レジスタファイル＃０　１３ａの構成例を示すブロック図である。

【図７】図４に示した実施例のプロセッサのタイミングチャートである。

【図８】第３の発明の動作説明図である。

【図９】第４の発明を適用した２スレッド並列実行型プロセッサの実施例のブロック図である。

【図１０】命令キュー＃０　２７ａの構成例を示すブロック図である。

【図１１】リオーダバッファ＃０　２４ａの構成例を示すブロック図である。

【図１２】レジスタデータセレクタ装置＃１　２６ｂの処理の論理を示すフローである。

【図１３】図９のプロセッサにおけるフォーク時の動作モデルを示す図である。

【図１４】第４の発明を適用した４スレッド並列実行２演算ユニット型プロセッサの実施例のブロック図である。

【図１５】第５の発明を適用した２スレッド並列実行型プロセッサの実施例のブロック図である。

【図１６】図１５のプロセッサにおけるフォーク時の動作モデルを示す図である。

【図１７】第６の発明の実施例のブロック図である。

【図１８】図１７の実施例のタイミングチャートである。

【図１９】第７の発明の実施例のブロック図である。

【図２０】第８の発明を適用した２スレッド並列実行型プロセッサの実施例のブロック図である。

【図２１】レジスタ写像テーブル７６ａの詳細なブロック図である。

【図２２】レジスタフリーテーブル８２の詳細なブロック図である。

【図２３】レジスタビジーテーブル８１の詳細なブロック図である。

【図２４】第１０の発明を適用した２スレッド並列実行型プロセッサの実施例のブロック図である。

【図２５】第１１の発明の実施例のモデルを示す図である。

【図２６】図１４の実施例に図２５の技術を適用した場合のレジスタファイル回りのブロック図である。

【図２７】本発明の他の実施の形態の実施例である２スレッド並列実行型のプロセッサのブロック図である。

【図２８】レジスタフリーテーブル１１３の詳細図である。

【図２９】本発明の他の実施の形態におけるレジスタフリーテーブルの詳細図である。

【図３０】従来のリオーダバッファ方式によるレジスタリネーミング機構の構成を示すブロック図である。

【図３１】従来のレジスタ写像テーブル方式によるレジスタリネーミング機構の構成を示すブロック図である。

【図３２】従来のマルチスレッド型のプロセッサの構成を示すブロック図である。

【図３３】Ｍｕｌｔｉｓｃａｌａｒ　Ｐｒｏｃｅｓｓｏｒの構成を示すブロック図である。

【符号の説明】

１…親スレッド（新スレッドを生成するスレッド）
２…スレッド生成命令（フォーク命令）
３…子スレッド（新スレッド）
４…スレッド管理手段
５ａ，５ｂ…スレッド実行手段
６ａ，６ｂ…レジスタファイル
７…レジスタ内容一括転送手段
８…共有メモリ
９ａ，９ｂ…スレッド実行手段
１０ａ，１０ｂ…命令キャッシュ
１１ａ，１１ｂ…命令デコーダ
１２ａ，１２ｂ…レジスタ選択手段
１３ａ，１３ｂ…レジスタファイル
１４ａ，１４ｂ…演算ユニット
１５…スレッド管理手段
１６…レジスタ選択テーブル
１７…レジスタ選択ビット
１８…レジスタ内容メモリ
１９…転送終了ビット
２０…更新ビット
２１ａ，２１ｂ…スレッド実行手段
２２ａ，２２ｂ…命令キャッシュ
２３ａ，２３ｂ…命令デコーダ
２４ａ，２４ｂ…リオーダバッファ
２５ａ，２５ｂ…レジスタファイル
２６ａ，２６ｂ…レジスタデータセレクタ装置
２７ａ，２７ｂ…命令キュー
２８ａ，２８ｂ…演算ユニット
２９…スレッド管理手段
３０…キューエントリ格納論理
３１…発行命令決定論理
３２…命令キューエントリ
３３…エントリ有効ビット
３４…レジスタ内容／リオーダバッファタグ格納メモリ
３５…レジスタ内容有効ビット
３６…結果書き込み場所指定タグ
３７…命令コード格納メモリ

29

３８…レジスタデータ供給決定論理
３９…エントリシフト制御論理
４０…リオーダバッファエントリ
４１…エントリ有効ビット
４２…結果書き込み指定タグ
４３…レジスタ番号
４４…スレッド生成ビット
４５…演算データ格納メモリ有効ビット
４６…演算データ格納メモリ
４７ａ，４７ｂ…結果振り分け装置
４８ａ，４８ｂ…レジスタデータ選択装置
４９ａ，４９ｂ…命令キュー
５０ａ，５０ｂ…スレッド実行手段
５１ａ，５１ｂ…命令キャッシュ
５２ａ，５２ｂ…命令デコーダ
５３ａ，５３ｂ…リオーダバッファ
５４ａ，５４ｂ…レジスタファイル
５５ａ，５５ｂ…レジスタデータセレクタ装置
５６ａ，５６ｂ…命令キュー
５７ａ，５７ｂ…演算ユニット
５８…スレッド管理手段
５９ａ，５９ｂ…レジスタファイル
６０…マルチプレクサ
６１…デバイダ
６２ａ，６２ｂ…参照許可ビット
６３ａ，６３ｂ…スレッド実行手段
６４ａ，６４ｂ…命令キャッシュ
６５ａ，６５ｂ…命令デコーダ
６６ａ，６６ｂ…リオーダバッファ
６７ａ，６７ｂ…レジスタファイル
６８ａ，６８ｂ…レジスタデータセレクタ装置
６９ａ，６９ｂ…命令キュー
７０ａ，７０ｂ…演算ユニット
７１…スレッド管理手段
７２…退避用レジスタファイル
７３ａ，７３ｂ…スレッド実行手段
７４ａ，７４ｂ…命令キャッシュ
７５ａ，７５ｂ…命令デコーダ
７６ａ，７６ｂ…レジスタ写像テーブル
７７ａ，７７ｂ…命令キュー
７８…物理レジスタファイル
７９ａ，７９ｂ…演算ユニット
８０ａ，８０ｂ…有効命令順序バッファ
８１…レジスタビジーテーブル
８２…レジスタフリーテーブル
８３…スレッド管理手段
８４…レジスタフリー決定論理
８５…フリーレジスタ検索論理
８６…状態テーブル
８７ａ，８７ｂ…スレッド実行手段

30

８８ａ，８８ｂ…命令キャッシュ
８９ａ，８９ｂ…命令デコーダ
９０ａ，９０ｂ…レジスタ写像テーブル
９１ａ，９１ｂ…命令キュー
９２…物理レジスタファイル
９３ａ，９３ｂ…演算ユニット
９４ａ，９４ｂ…有効命令順序バッファ
９５…レジスタビジーテーブル
９６…レジスタフリーテーブル
９７…スレッド管理手段
９８…退避用レジスタ写像テーブル
９９ａ，９９ｂ，９９ｃ，９９ｄ…レジスタファイル
１００ａ，１００ｂ…結果振り分け装置
１０１ａ，１０１ｂ…レジスタデータ選択装置
１０２ａ，１０２ｂ，１０２ｃ，１０２ｄ…リオーダバッファ
１０３…リング状レジスタファイル転送手段
１０４ａ，１０４ｂ…スレッド実行手段
１０５ａ，１０５ｂ…命令キャッシュ
１０６ａ，１０６ｂ…命令デコーダ
１０７ａ，１０７ｂ…レジスタ写像テーブル
１０８ａ，１０８ｂ…命令キュー
１０９…物理レジスタファイル
１１０ａ，１１０ｂ…演算ユニット
１１１ａ，１１１ｂ…有効命令順序バッファ
１１２…レジスタビジーテーブル
１１３…レジスタフリーテーブル
１１４…スレッド管理手段
１１５…レジスタフリー決定論理
１１６…フリーレジスタ検索論理
１１７…状態テーブル
１１８…レジスタフリー決定論理
１１９…フリーレジスタ検索論理
１２０…状態テーブル
１２１…レジスタ使用中ビット
１２２…レジスタファイル
１２３…リオーダ・バッファ
１２４…レジスタ写像テーブル
１２５…レジスタファイル
１２６…有効命令順序バッファ
１２７…レジスタビジーテーブル
１２８…レジスタフリーテーブル
１２９…命令取得装置
１３０…命令解読装置
１３１…機能実行装置
１３２…命令依存解析装置
１３３…命令調停装置
１３４…シーケンサ
１３５…プロセッシングユニット
１３６…結合ネットワーク

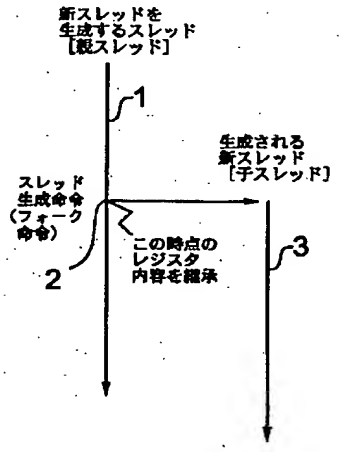
31

１３７…データバンク
１３８…命令キャッシュ
１３９…実行ユニット
１４０…レジスタファイル

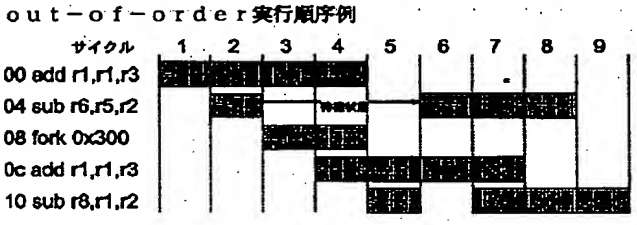
32

１４１…ＡＲＢ（Ａｄｄｒｅｓｓ　Ｒｅｓｏｌｕｔｉｏｎ　Ｂｕｆｆｅｒ）
１４２…データキャッシュ
１５０…レジスタ内容転送手段

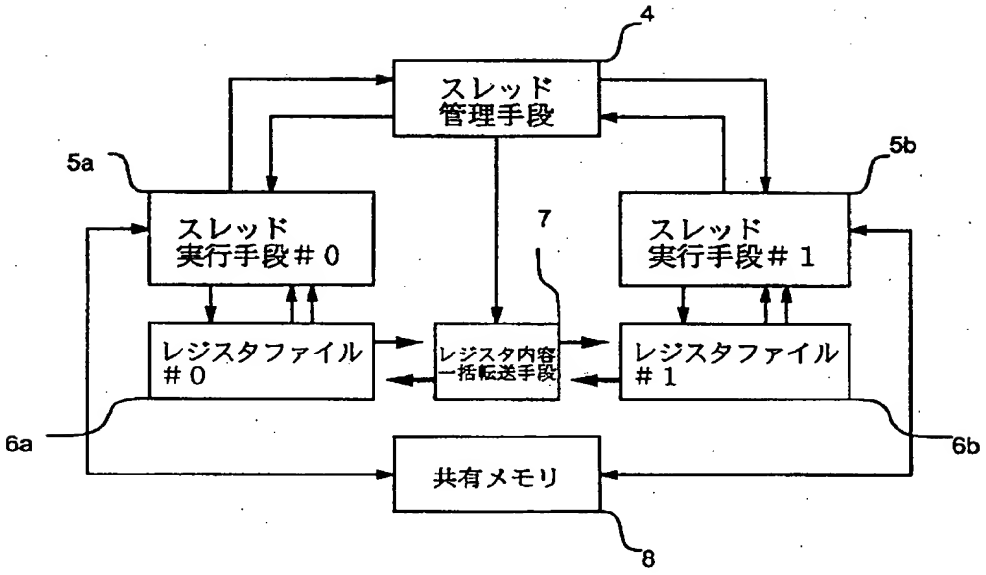
【図１】



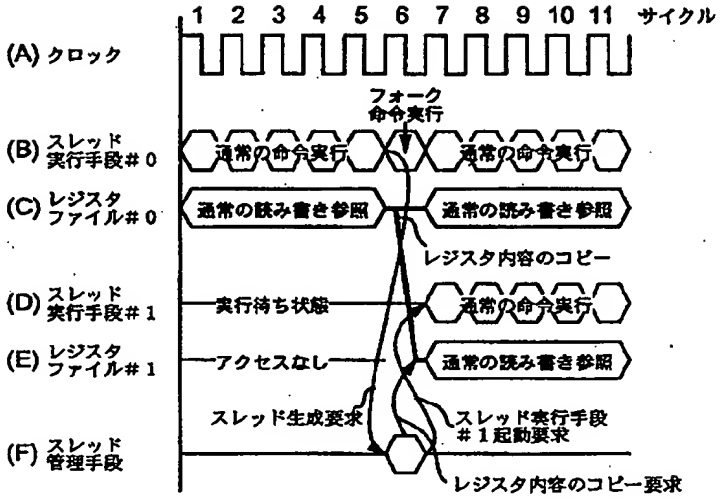
【図８】



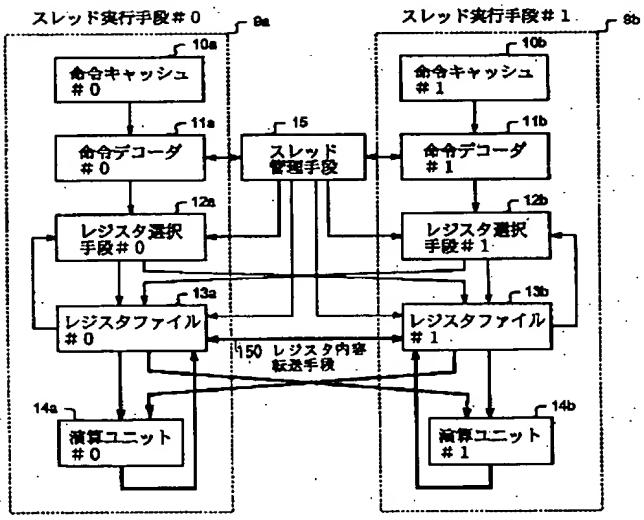
【図２】



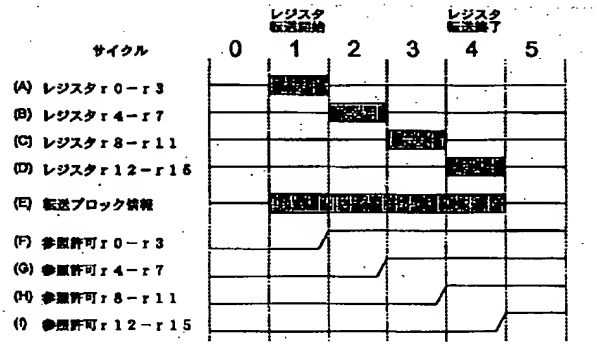
【図３】


1 2 3 4 5 6 7 8 9 10 11 サイクル
(A) クロック
フォーク
命令実行
(B) スレッド
実行手段＃０
通常の命令実行
通常の命令実行
(C) レジスタ
ファイル＃０
通常の読み書き参照
通常の読み書き参照
レジスタ内容のコピー
(D) スレッド
実行手段＃１
実行待ち状態
通常の命令実行
(E) レジスタ
ファイル＃１
アクセスなし
通常の読み書き参照
スレッド生成要求
スレッド実行手段
＃１起動要求
(F) スレッド
管理手段
レジスタ内容のコピー要求

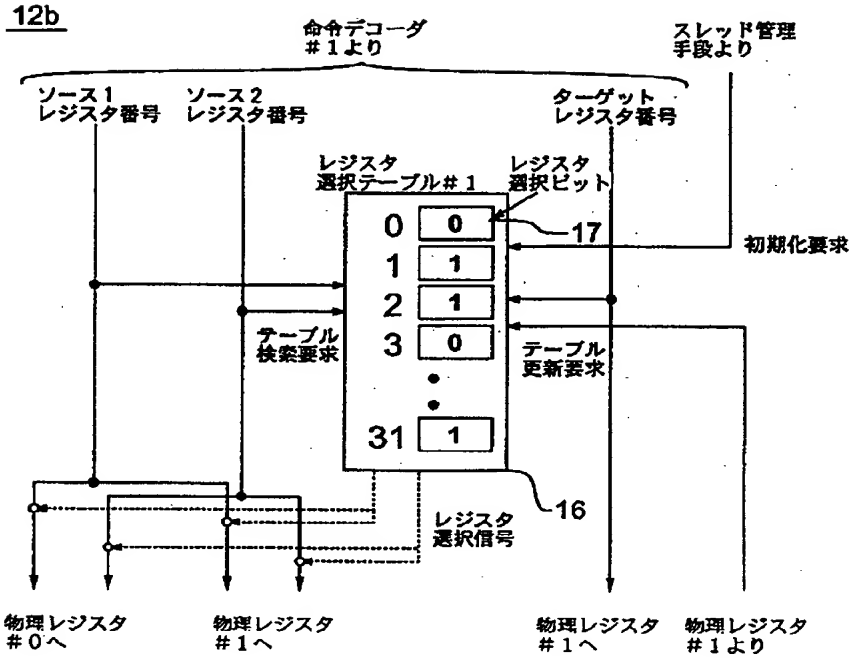
【図４】


スレッド実行手段＃０
8a
スレッド実行手段＃１
8b
10a
命令キャッシュ
＃０
10b
命令キャッシュ
＃１
11a
命令デコーダ
＃０
15
スレッド
管理手段
11b
命令デコーダ
＃１
12a
レジスタ選択
手段＃０
12b
レジスタ選択
手段＃１
13a
レジスタファイル
＃０
13b
レジスタファイル
＃１
150 レジスタ内容
転送手段
14a
演算ユニット
＃０
14b
演算ユニット
＃１

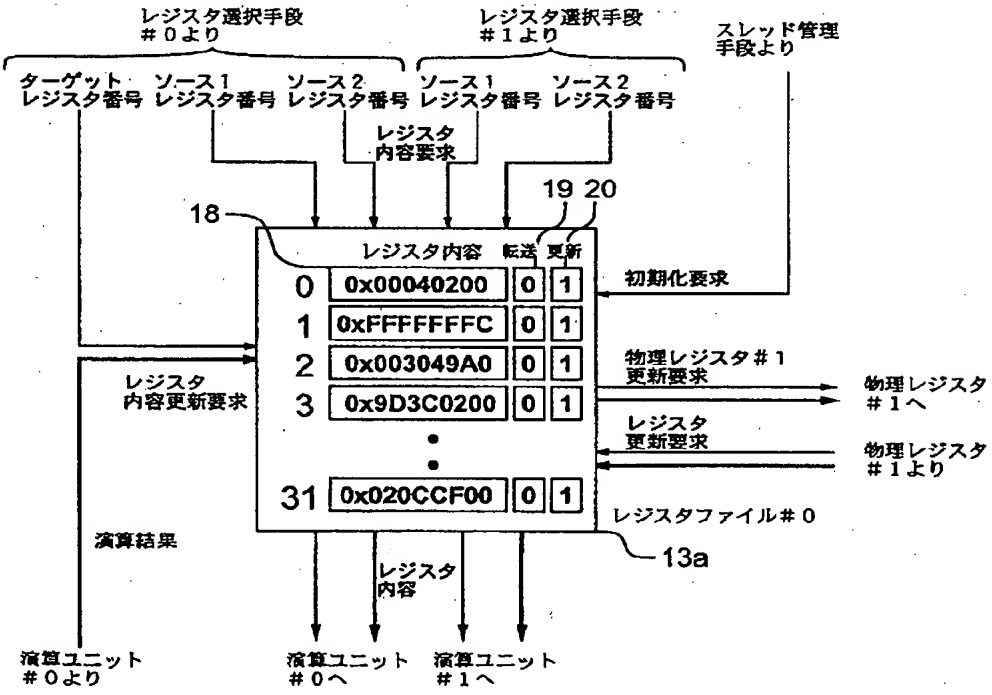
【図１８】


サイクル
0 1 2 3 4 5
(A) レジスタｒ０－ｒ３
(B) レジスタｒ４－ｒ７
(C) レジスタｒ８－ｒ１１
(D) レジスタｒ１２－ｒ１５
(E) 転送ブロック情報
(F) 参照許可ｒ０－ｒ３
(G) 参照許可ｒ４－ｒ７
(H) 参照許可ｒ８－ｒ１１
(I) 参照許可ｒ１２－ｒ１５

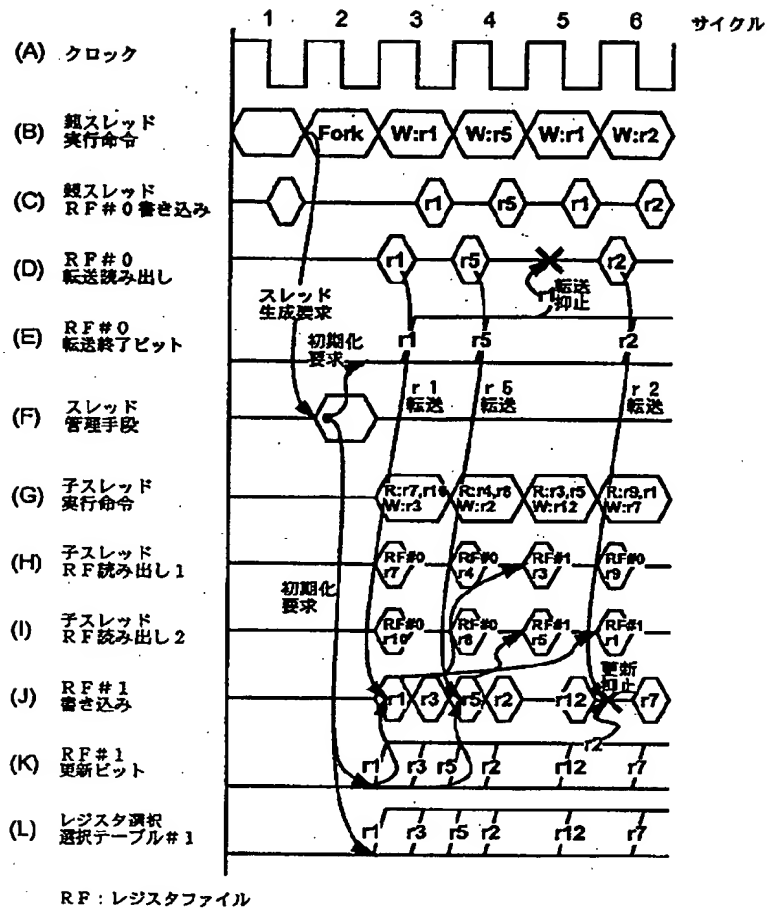
【図５】


12b
命令デコーダ
＃１より
スレッド管理
手段より
ソース１
レジスタ番号
ソース２
レジスタ番号
ターゲット
レジスタ番号
レジスタ
選択テーブル＃１
レジスタ
選択ビット
0
1
2
3
31
0
1
1
0
1
17
初期化要求
テーブル
検索要求
テーブル
更新要求
16
レジスタ
選択信号
物理レジスタ
＃０へ
物理レジスタ
＃１へ
物理レジスタ
＃１へ
物理レジスタ
＃１より

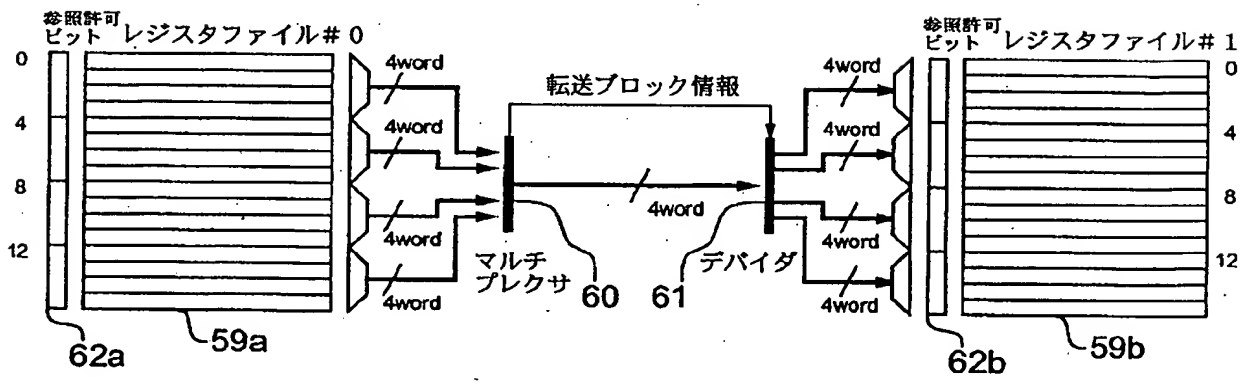
【図６】


レジスタ選択手段
＃０より
レジスタ選択手段
＃１より
スレッド管理
手段より
ターゲット
レジスタ番号
ソース１
レジスタ番号
ソース２
レジスタ番号
ソース１
レジスタ番号
ソース２
レジスタ番号
レジスタ
内容要求
18
19 20
レジスタ内容
転送
更新
0
1
2
3
31
0x00040200
0xFFFFFFFC
0x003049A0
0x9D3C0200
0x020CCF00
0
1
初期化要求
物理レジスタ＃１
更新要求
物理レジスタ
＃１へ
レジスタ
更新要求
物理レジスタ
＃１より
レジスタ
内容更新要求
演算結果
レジスタファイル＃０
13a
レジスタ
内容
演算ユニット
＃０より
演算ユニット
＃０へ
演算ユニット
＃１へ

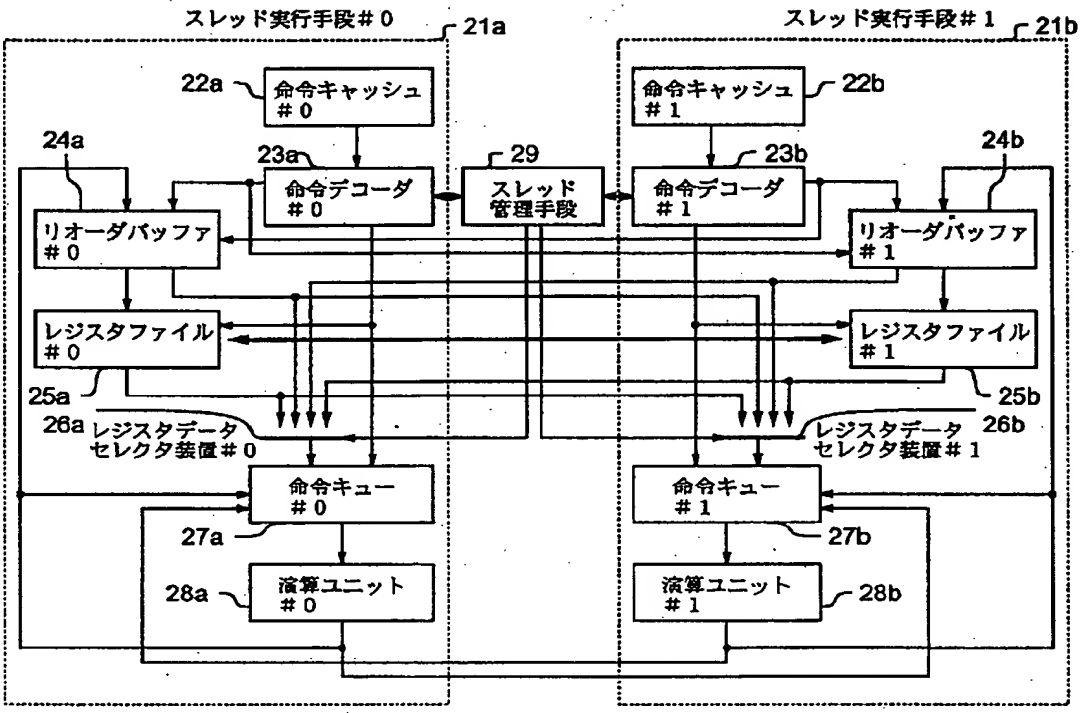
【図７】


1
2
3
4
5
6
サイクル
(A) クロック
(B) 親スレッド 実行命令
(C) 親スレッド ＲＦ＃０書き込み
(D) ＲＦ＃０ 転送読み出し
(E) ＲＦ＃０ 転送終了ビット
(F) スレッド 管理手段
(G) 子スレッド 実行命令
(H) 子スレッド ＲＦ読み出し１
(I) 子スレッド ＲＦ読み出し２
(J) ＲＦ＃１ 書き込み
(K) ＲＦ＃１ 更新ビット
(L) レジスタ選択 選択テーブル＃１
Fork
W:r1
W:r5
W:r1
W:r2
スレッド 生成要求
初期化 要求
転送 抑止
r 1 転送
r 5 転送
r 2 転送
更新 抑止
ＲＦ：レジスタファイル

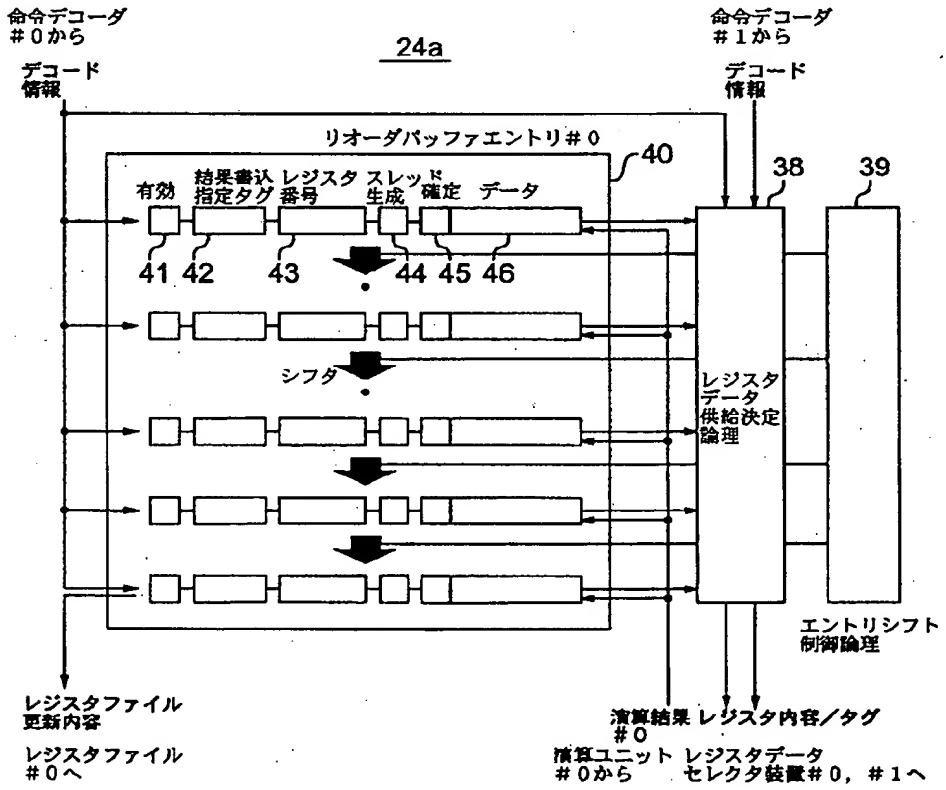
【図１７】


参照許可 ビット
レジスタファイル＃０
0
4
8
12
4word
転送ブロック情報
マルチ プレクサ
60
61
デバイダ
参照許可 ビット
レジスタファイル＃１
62a
59a
62b
59b

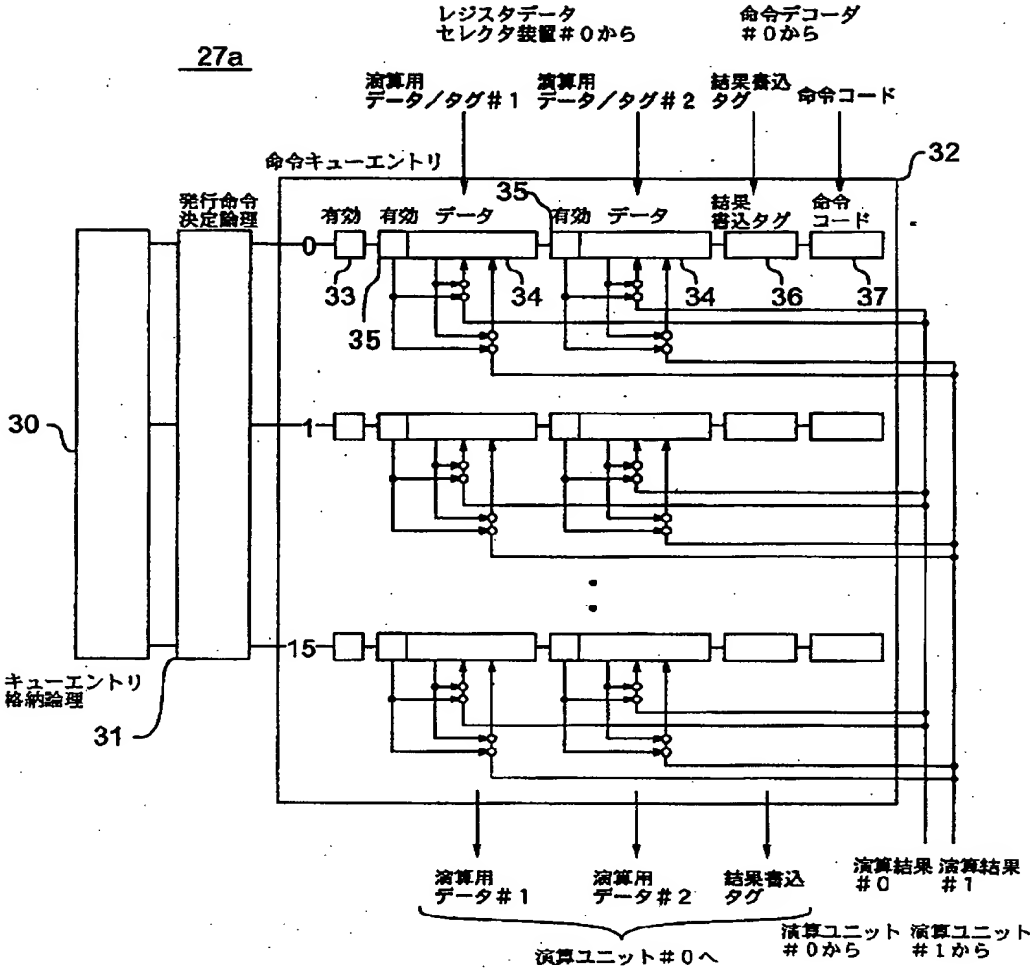
【図９】


スレッド実行手段＃０
21a
スレッド実行手段＃１
21b
22a
命令キャッシュ＃０
命令キャッシュ＃１
22b
24a
24b
23a
29
23b
命令デコーダ＃０
スレッド管理手段
命令デコーダ＃１
リオーダバッファ＃０
リオーダバッファ＃１
レジスタファイル＃０
レジスタファイル＃１
25a
25b
26a
レジスタデータセレクタ装置＃０
レジスタデータセレクタ装置＃１
26b
命令キュー＃０
命令キュー＃１
27a
27b
28a
演算ユニット＃０
演算ユニット＃１
28b

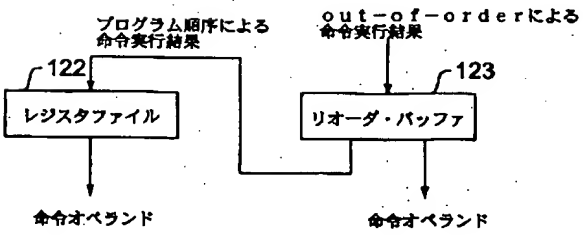
【図１１】


命令デコーダ＃０から
デコード情報
24a
命令デコーダ＃１から
デコード情報
リオーダバッファエントリ＃０
40
38
39
有効
結果書込指定タグ
レジスタ番号
スレッド生成
確定
データ
41
42
43
44
45
46
シフタ
レジスタデータ供給決定論理
エントリシフト制御論理
レジスタファイル更新内容
レジスタファイル＃０へ
演算結果＃０
レジスタ内容／タグ
演算ユニット＃０から
レジスタデータセレクタ装置＃０，＃１へ

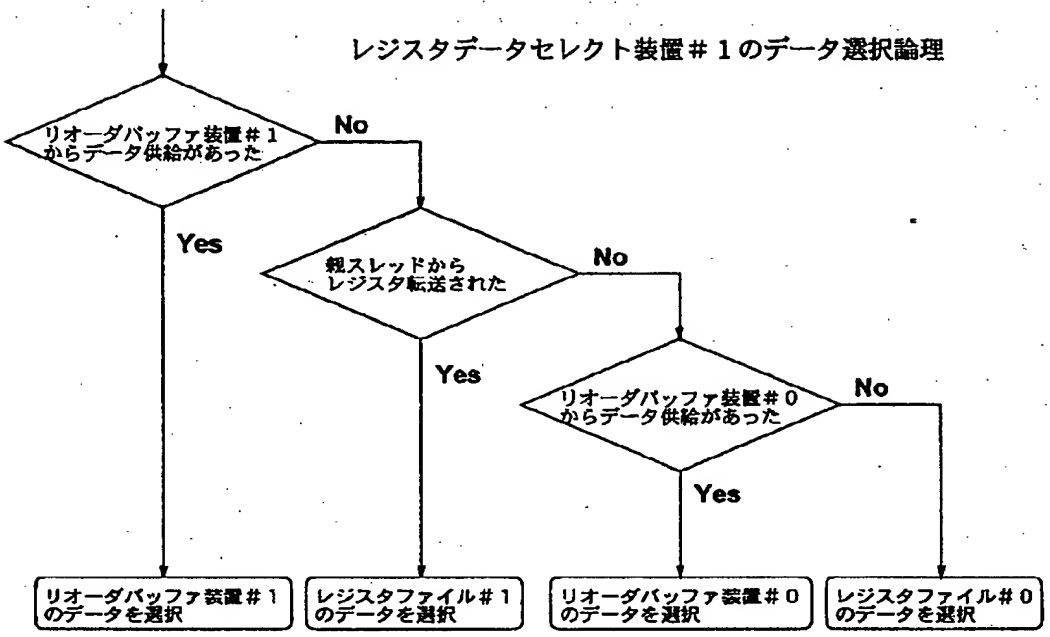
【図１０】


27a
レジスタデータ
セレクタ装置＃０から
命令デコーダ
＃０から
演算用
データ／タグ＃１
演算用
データ／タグ＃２
結果書込
タグ
命令コード
命令キューエントリ
32
発行命令
決定論理
35
有効
有効
データ
有効
データ
結果
書込タグ
命令
コード
0
33
34
34
36
37
35
30
1
15
キューエントリ
格納論理
31
演算用
データ＃１
演算用
データ＃２
結果書込
タグ
演算結果
＃０
演算結果
＃１
演算ユニット
＃０から
演算ユニット
＃１から
演算ユニット＃０へ

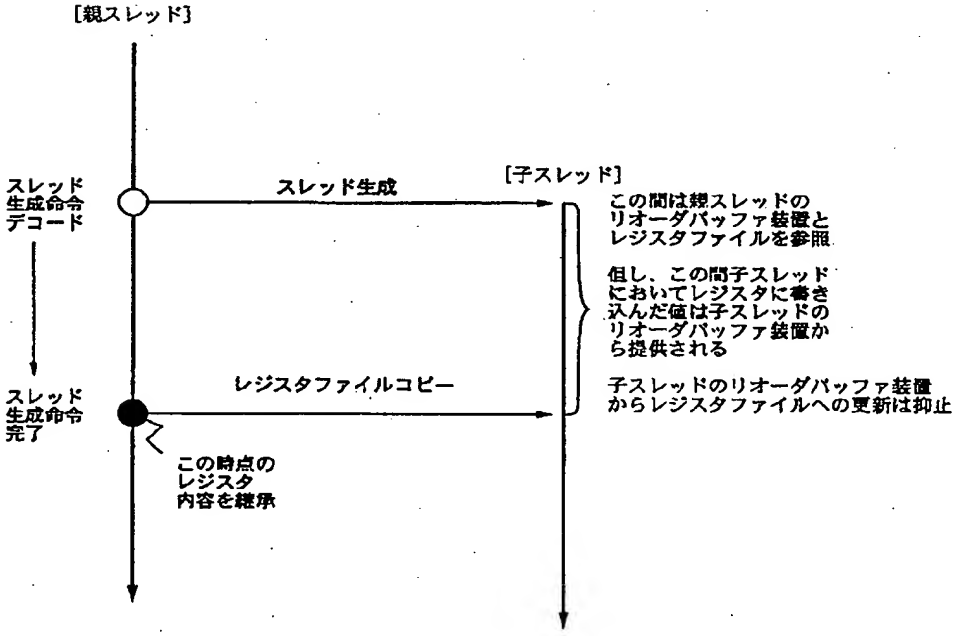
【図３０】


プログラム順序による
命令実行結果
out－of－orderによる
命令実行結果
122
123
レジスタファイル
リオーダ・バッファ
命令オペランド
命令オペランド

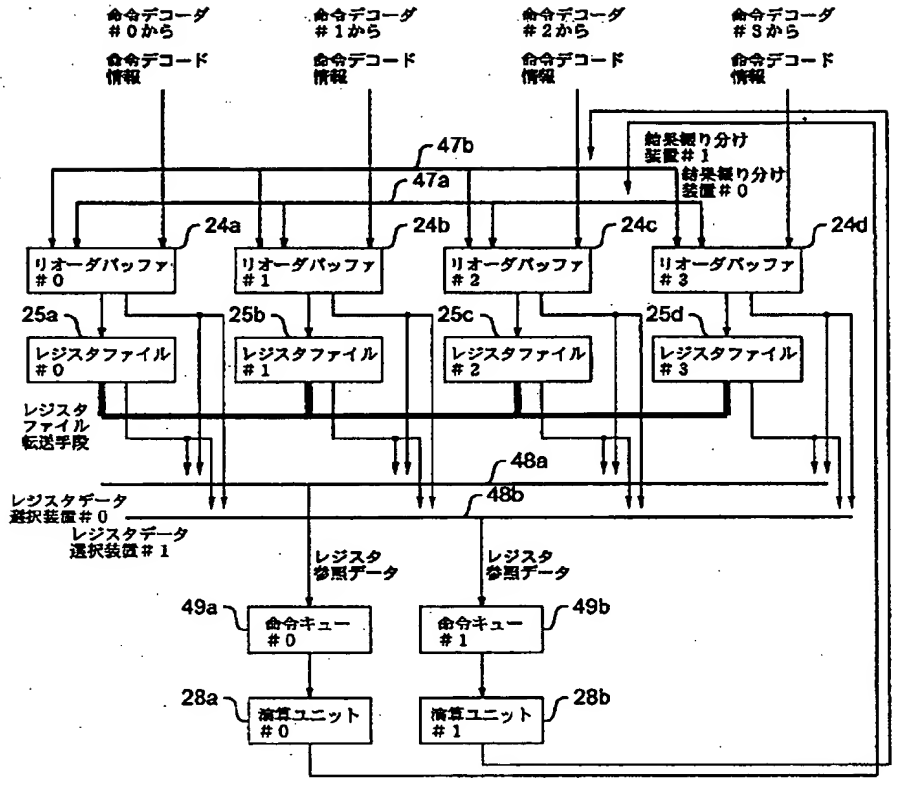
【図１２】


レジスタデータセレクト装置＃１のデータ選択論理
リオーダバッファ装置＃１からデータ供給があった
No
Yes
親スレッドからレジスタ転送された
No
Yes
リオーダバッファ装置＃０からデータ供給があった
No
Yes
リオーダバッファ装置＃１のデータを選択
レジスタファイル＃１のデータを選択
リオーダバッファ装置＃０のデータを選択
レジスタファイル＃０のデータを選択

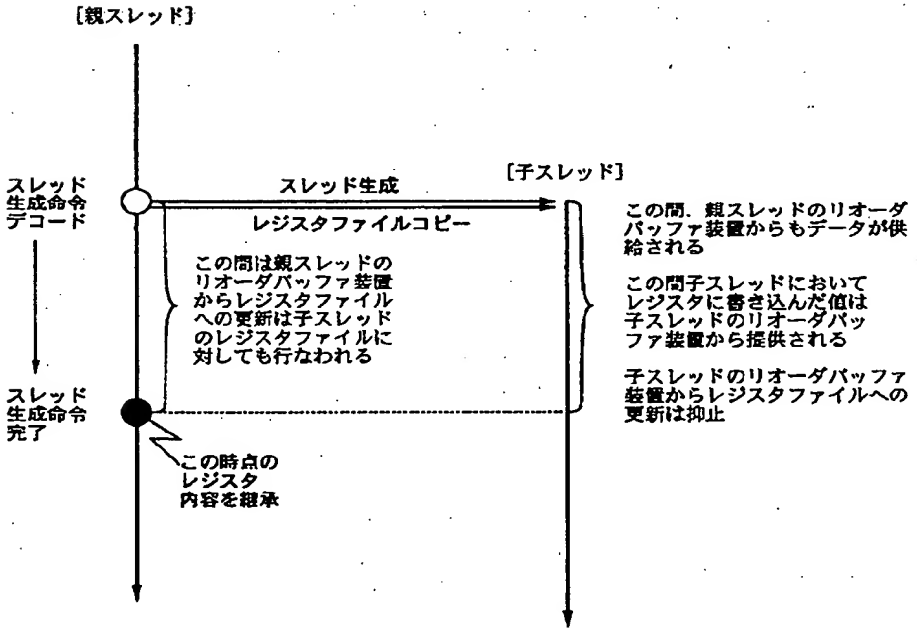
【図１３】


[親スレッド]
スレッド生成命令デコード
スレッド生成
[子スレッド]
この間は親スレッドのリオーダバッファ装置とレジスタファイルを参照
但し、この間子スレッドにおいてレジスタに書き込んだ値は子スレッドのリオーダバッファ装置から提供される
子スレッドのリオーダバッファ装置からレジスタファイルへの更新は抑止
スレッド生成命令完了
レジスタファイルコピー
この時点のレジスタ内容を継承

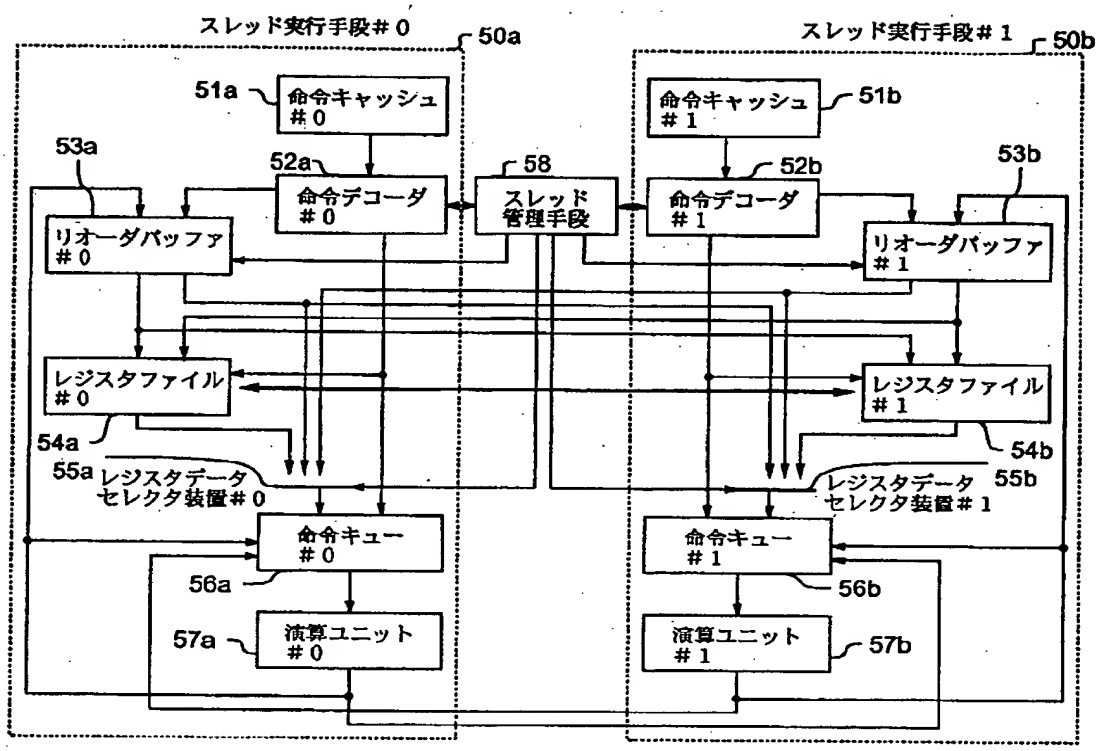
【図１４】


命令デコーダ
＃０から
命令デコーダ
＃１から
命令デコーダ
＃２から
命令デコーダ
＃３から
命令デコード
情報
命令デコード
情報
命令デコード
情報
命令デコード
情報
結果振り分け
装置＃１
結果振り分け
装置＃０
47b
47a
24a
24b
24c
24d
リオーダバッファ
＃０
リオーダバッファ
＃１
リオーダバッファ
＃２
リオーダバッファ
＃３
25a
25b
25c
25d
レジスタファイル
＃０
レジスタファイル
＃１
レジスタファイル
＃２
レジスタファイル
＃３
レジスタ
ファイル
転送手段
48a
48b
レジスタデータ
選択装置＃０
レジスタデータ
選択装置＃１
レジスタ
参照データ
レジスタ
参照データ
49a
命令キュー
＃０
命令キュー
＃１
49b
28a
演算ユニット
＃０
演算ユニット
＃１
28b

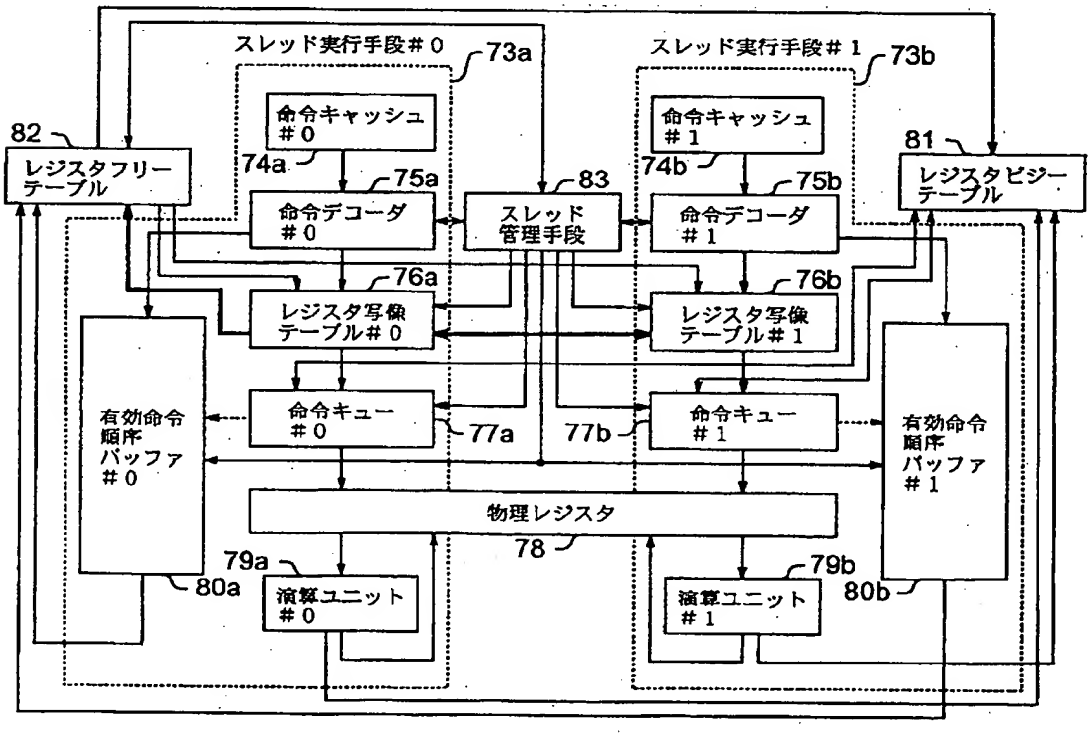
【図１６】


[親スレッド]
[子スレッド]
スレッド
生成命令
デコード
スレッド生成
レジスタファイルコピー
この間は親スレッドの
リオーダバッファ装置
からレジスタファイル
への更新は子スレッド
のレジスタファイルに
対しても行なわれる
この間、親スレッドのリオーダ
バッファ装置からもデータが供
給される
この間子スレッドにおいて
レジスタに書き込んだ値は
子スレッドのリオーダバッ
ファ装置から提供される
子スレッドのリオーダバッファ
装置からレジスタファイルへの
更新は抑止
スレッド
生成命令
完了
この時点の
レジスタ
内容を継承

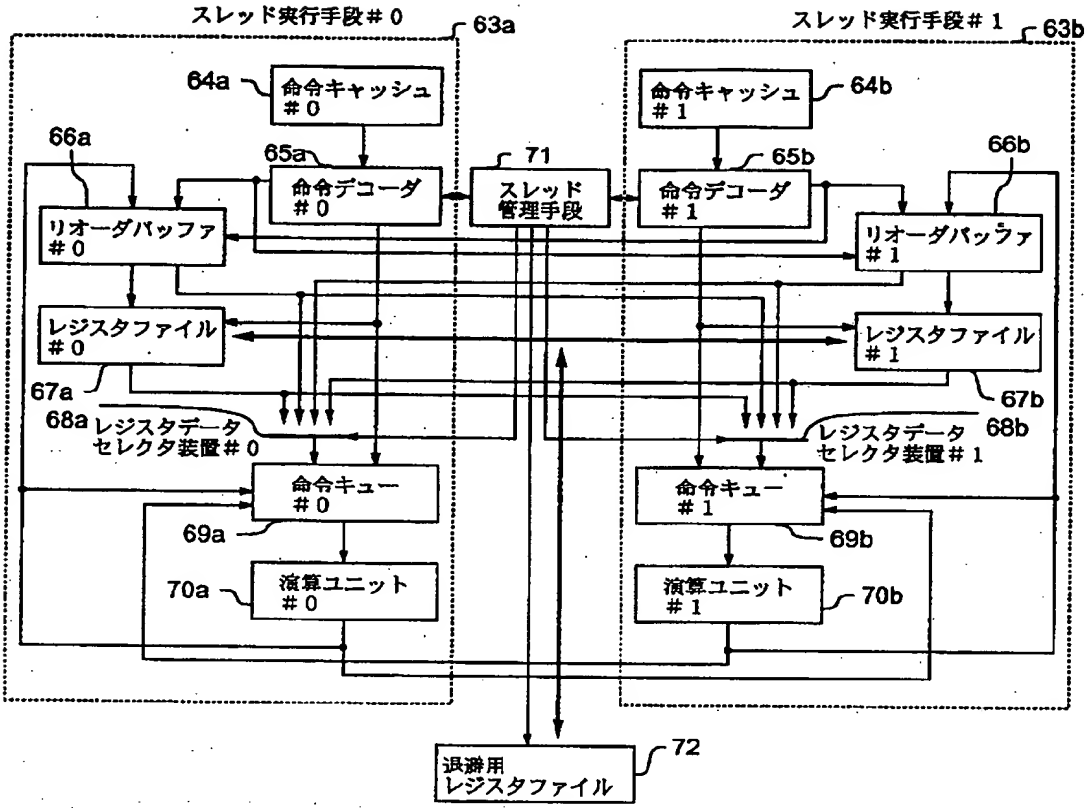
【図１５】


スレッド実行手段＃０
50a
スレッド実行手段＃１
50b
51a
命令キャッシュ＃０
命令キャッシュ＃１
51b
53a
52a
命令デコーダ＃０
58
スレッド管理手段
命令デコーダ＃１
52b
53b
リオーダバッファ＃０
リオーダバッファ＃１
レジスタファイル＃０
レジスタファイル＃１
54a
54b
55a
レジスタデータセレクタ装置＃０
レジスタデータセレクタ装置＃１
55b
命令キュー＃０
命令キュー＃１
56a
56b
57a
演算ユニット＃０
演算ユニット＃１
57b

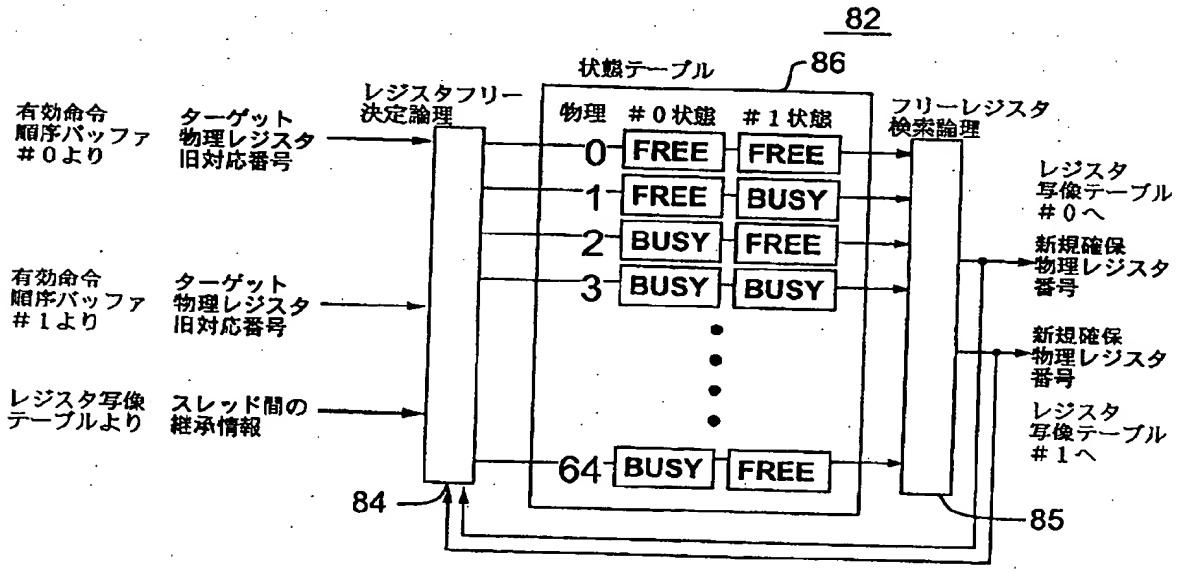
【図２０】


スレッド実行手段＃０
73a
スレッド実行手段＃１
73b
命令キャッシュ＃０
命令キャッシュ＃１
82
81
レジスタフリーテーブル
レジスタビジーテーブル
74a
74b
75a
83
75b
命令デコーダ＃０
スレッド管理手段
命令デコーダ＃１
76a
76b
レジスタ写像テーブル＃０
レジスタ写像テーブル＃１
有効命令順序バッファ＃０
命令キュー＃０
77a
77b
命令キュー＃１
有効命令順序バッファ＃１
物理レジスタ
78
79a
79b
80a
80b
演算ユニット＃０
演算ユニット＃１

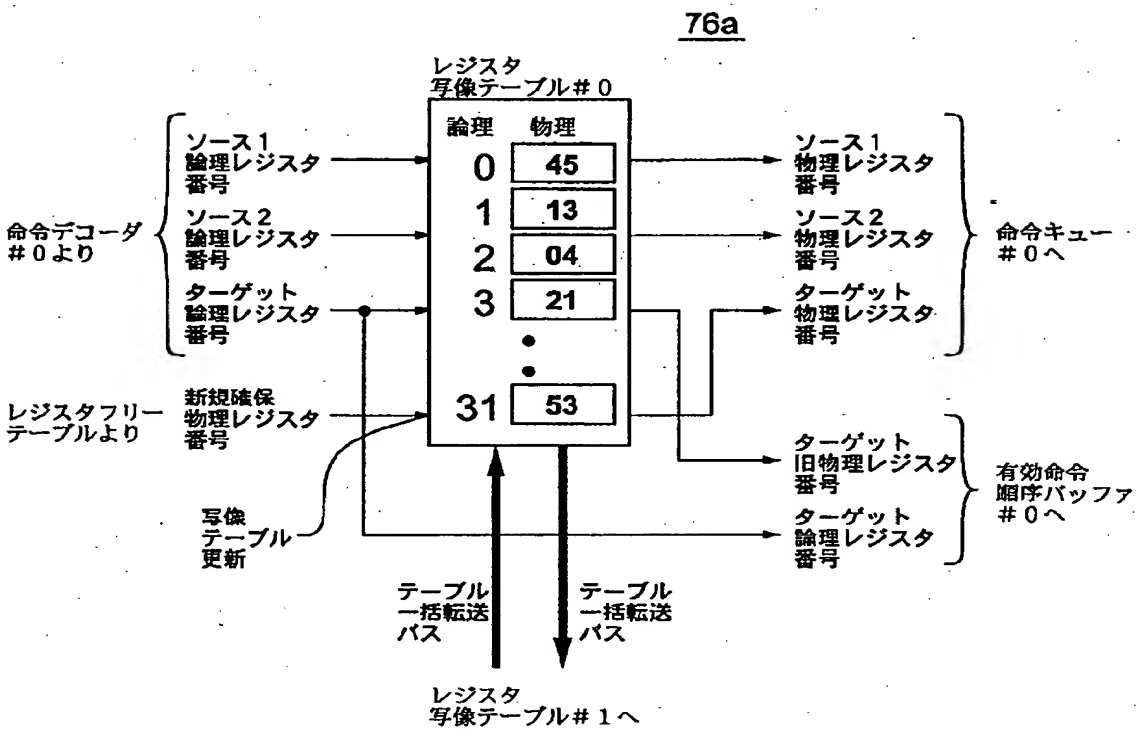
【図１９】


スレッド実行手段＃０
63a
スレッド実行手段＃１
63b
64a
命令キャッシュ＃０
命令キャッシュ＃１
64b
66a
65a
71
スレッド管理手段
65b
66b
命令デコーダ＃０
命令デコーダ＃１
リオーダバッファ＃０
リオーダバッファ＃１
レジスタファイル＃０
レジスタファイル＃１
67a
67b
68a
レジスタデータセレクタ装置＃０
レジスタデータセレクタ装置＃１
68b
命令キュー＃０
命令キュー＃１
69a
69b
演算ユニット＃０
演算ユニット＃１
70a
70b
退避用レジスタファイル
72

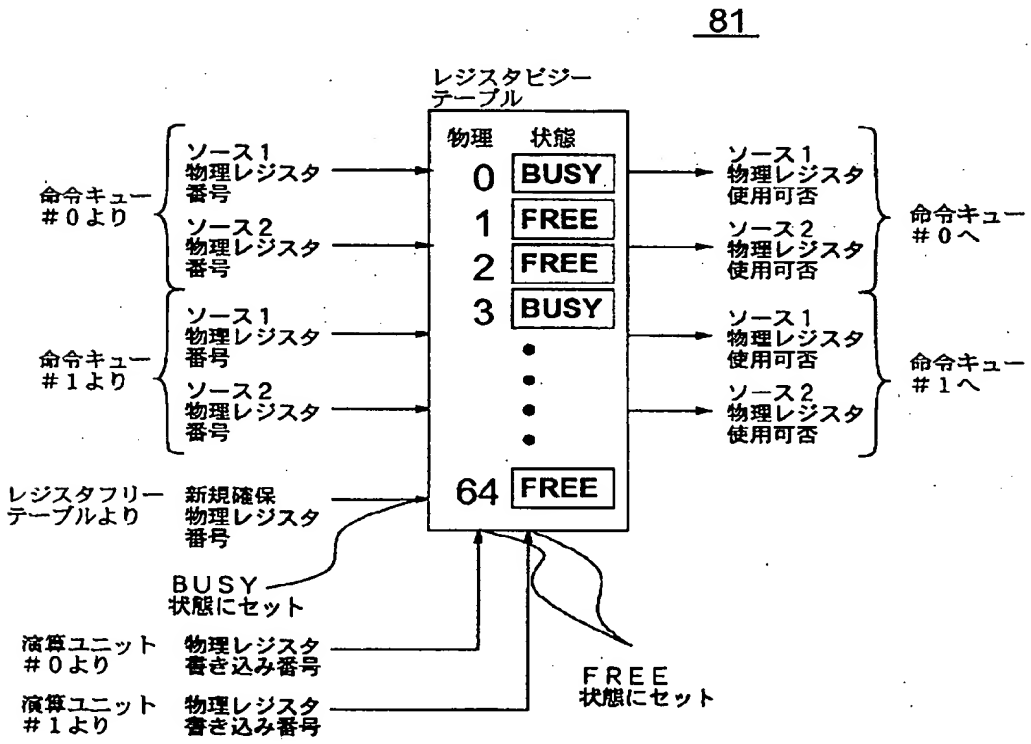
【図２２】


82
状態テーブル
86
レジスタフリー決定論理
物理 ＃０状態 ＃１状態
フリーレジスタ検索論理
有効命令順序バッファ＃０より
ターゲット物理レジスタ旧対応番号
0 FREE FREE
1 FREE BUSY
2 BUSY FREE
3 BUSY BUSY
64 BUSY FREE
レジスタ写像テーブル＃０へ
新規確保物理レジスタ番号
有効命令順序バッファ＃１より
ターゲット物理レジスタ旧対応番号
新規確保物理レジスタ番号
レジスタ写像テーブルより
スレッド間の継承情報
レジスタ写像テーブル＃１へ
84
85

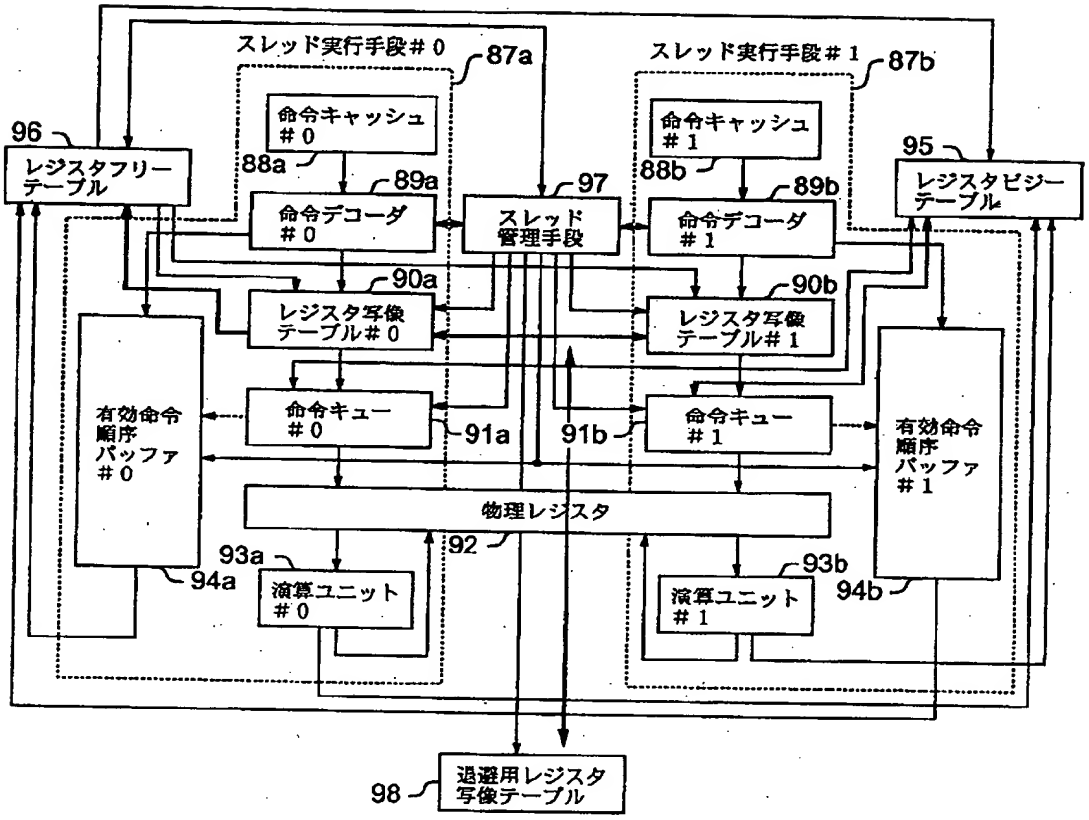
【図２１】


76a
レジスタ
写像テーブル＃０
論理
物理
0
45
1
13
2
04
3
21
31
53
ソース１
論理レジスタ
番号
ソース２
論理レジスタ
番号
ターゲット
論理レジスタ
番号
命令デコーダ
＃０より
レジスタフリー
テーブルより
新規確保
物理レジスタ
番号
写像
テーブル
更新
ソース１
物理レジスタ
番号
ソース２
物理レジスタ
番号
ターゲット
物理レジスタ
番号
命令キュー
＃０へ
ターゲット
旧物理レジスタ
番号
ターゲット
論理レジスタ
番号
有効命令
順序バッファ
＃０へ
テーブル
一括転送
バス
テーブル
一括転送
バス
レジスタ
写像テーブル＃１へ

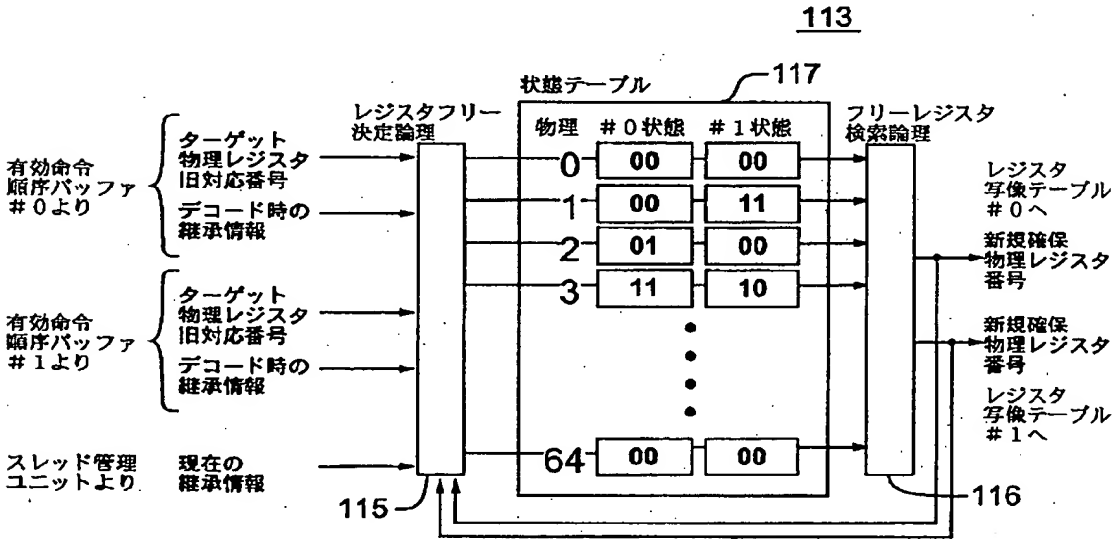
【図２３】


81
レジスタビジー
テーブル
物理
状態
0
BUSY
1
FREE
2
FREE
3
BUSY
64
FREE
命令キュー
＃０より
ソース１
物理レジスタ
番号
ソース２
物理レジスタ
番号
命令キュー
＃１より
ソース１
物理レジスタ
番号
ソース２
物理レジスタ
番号
レジスタフリー
テーブルより
新規確保
物理レジスタ
番号
BUSY
状態にセット
演算ユニット
＃０より
物理レジスタ
書き込み番号
演算ユニット
＃１より
物理レジスタ
書き込み番号
ソース１
物理レジスタ
使用可否
ソース２
物理レジスタ
使用可否
命令キュー
＃０へ
ソース１
物理レジスタ
使用可否
ソース２
物理レジスタ
使用可否
命令キュー
＃１へ
FREE
状態にセット

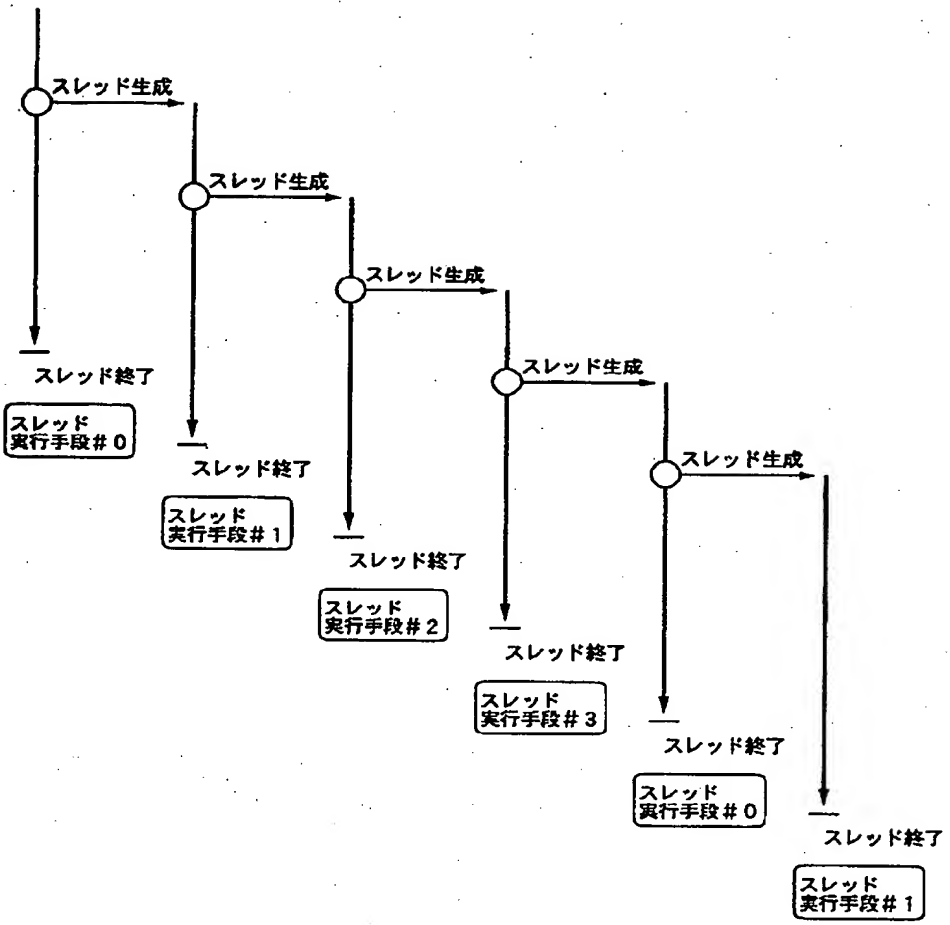
【図２４】


スレッド実行手段＃０
87a
スレッド実行手段＃１
87b
命令キャッシュ＃０
命令キャッシュ＃１
96
レジスタフリーテーブル
95
レジスタビジーテーブル
88a
88b
89a
89b
97
命令デコーダ＃０
スレッド管理手段
命令デコーダ＃１
90a
90b
レジスタ写像テーブル＃０
レジスタ写像テーブル＃１
有効命令順序バッファ＃０
命令キュー＃０
命令キュー＃１
有効命令順序バッファ＃１
91a
91b
物理レジスタ
92
93a
93b
94a
94b
演算ユニット＃０
演算ユニット＃１
98
退避用レジスタ写像テーブル

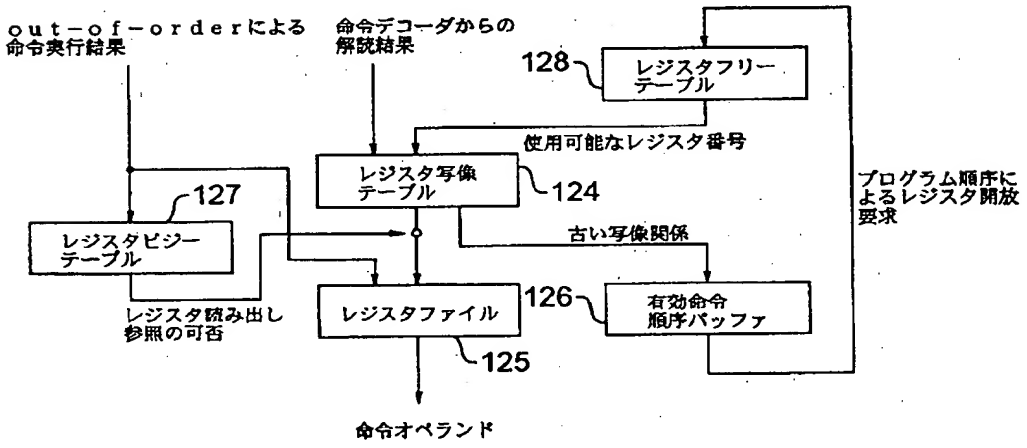
【図２８】


113
状態テーブル
117
レジスタフリー決定論理
物理
＃０状態
＃１状態
フリーレジスタ検索論理
有効命令順序バッファ＃０より
ターゲット物理レジスタ旧対応番号
デコード時の継承情報
有効命令順序バッファ＃１より
スレッド管理ユニットより
現在の継承情報
0
1
2
3
64
00
11
01
10
レジスタ写像テーブル＃０へ
新規確保物理レジスタ番号
レジスタ写像テーブル＃１へ
115
116

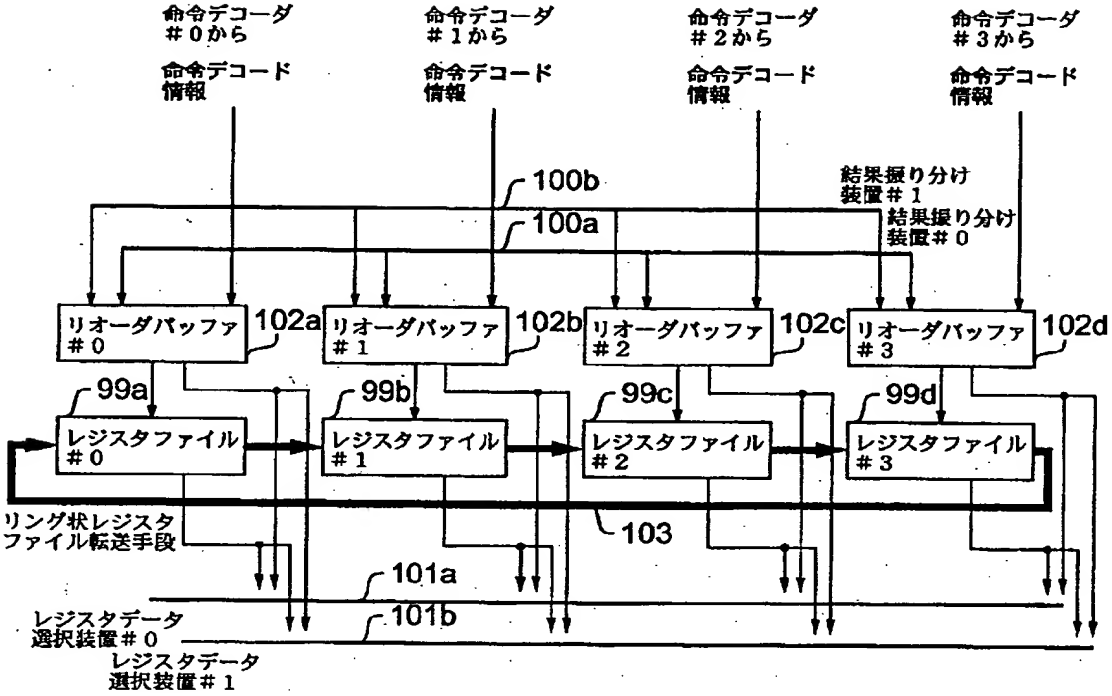
【図２５】


スレッド生成
スレッド生成
スレッド生成
スレッド生成
スレッド生成
スレッド終了
スレッド
実行手段＃０
スレッド終了
スレッド
実行手段＃１
スレッド終了
スレッド
実行手段＃２
スレッド終了
スレッド
実行手段＃３
スレッド終了
スレッド
実行手段＃０
スレッド終了
スレッド
実行手段＃１

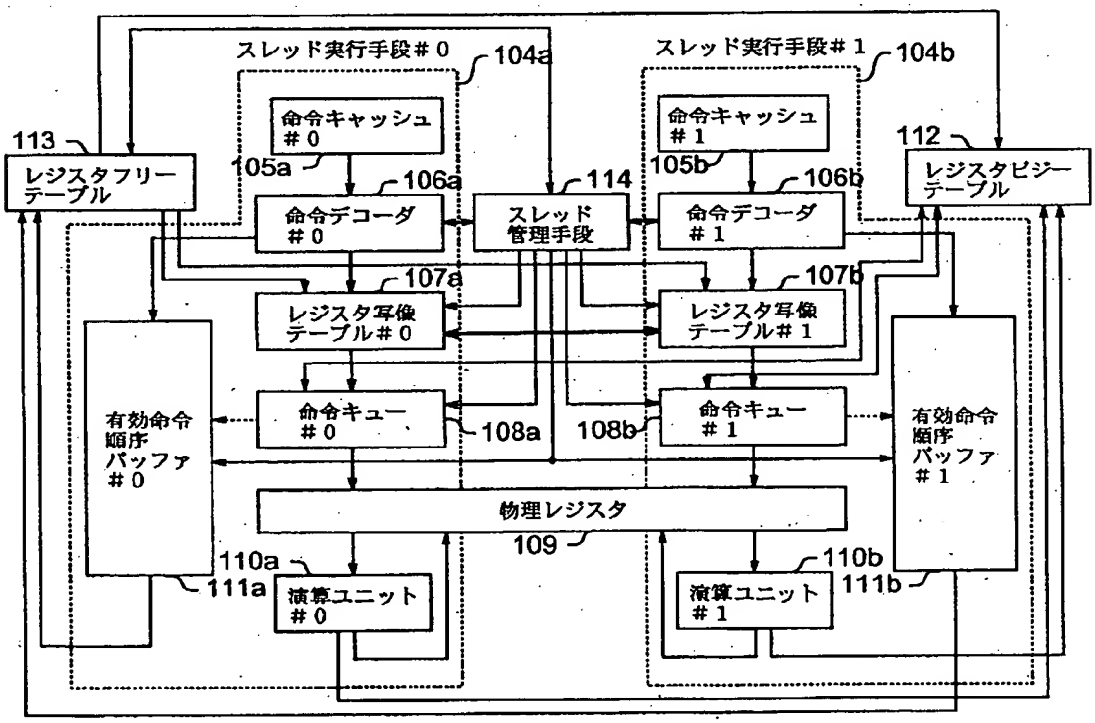
【図３１】


ｏｕｔ－ｏｆ－ｏｒｄｅｒによる
命令実行結果
命令デコーダからの
解読結果
128
レジスタフリー
テーブル
使用可能なレジスタ番号
レジスタ写像
テーブル
124
プログラム順序に
よるレジスタ開放
要求
127
レジスタビジー
テーブル
古い写像関係
レジスタ読み出し
参照の可否
レジスタファイル
126
有効命令
順序バッファ
125
命令オペランド

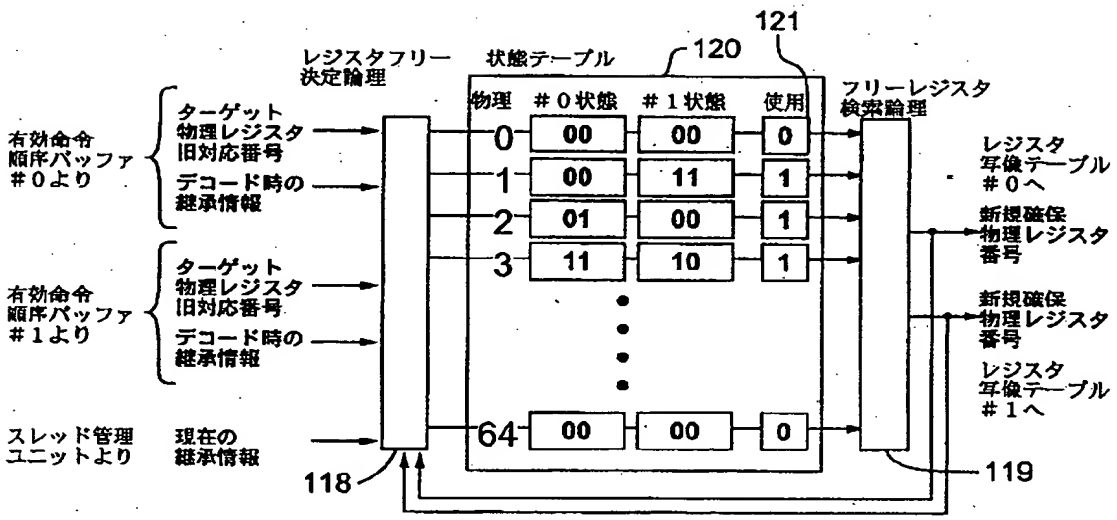
【図２６】


命令デコーダ
＃０から
命令デコード
情報
命令デコーダ
＃１から
命令デコード
情報
命令デコーダ
＃２から
命令デコード
情報
命令デコーダ
＃３から
命令デコード
情報
100b
100a
結果振り分け
装置＃１
結果振り分け
装置＃０
リオーダバッファ
＃０
102a
リオーダバッファ
＃１
102b
リオーダバッファ
＃２
102c
リオーダバッファ
＃３
102d
99a
99b
99c
99d
レジスタファイル
＃０
レジスタファイル
＃１
レジスタファイル
＃２
レジスタファイル
＃３
リング状レジスタ
ファイル転送手段
103
101a
101b
レジスタデータ
選択装置＃０
レジスタデータ
選択装置＃１

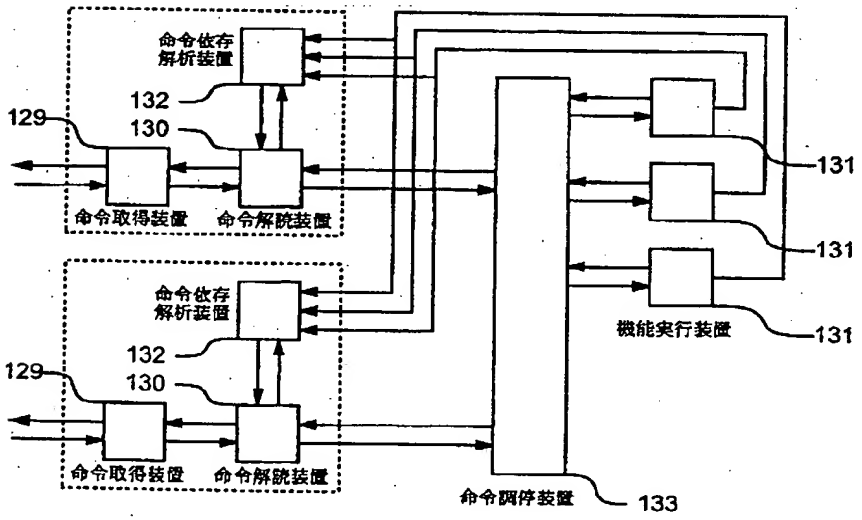
【図２７】


スレッド実行手段＃０
104a
スレッド実行手段＃１
104b
命令キャッシュ
＃０
命令キャッシュ
＃１
113
レジスタフリー
テーブル
105a
105b
112
レジスタビジー
テーブル
106a
114
106b
命令デコーダ
＃０
スレッド
管理手段
命令デコーダ
＃１
107a
107b
レジスタ写像
テーブル＃０
レジスタ写像
テーブル＃１
有効命令
順序
バッファ
＃０
命令キュー
＃０
108a
108b
命令キュー
＃１
有効命令
順序
バッファ
＃１
物理レジスタ
109
110a
111a
演算ユニット
＃０
110b
111b
演算ユニット
＃１

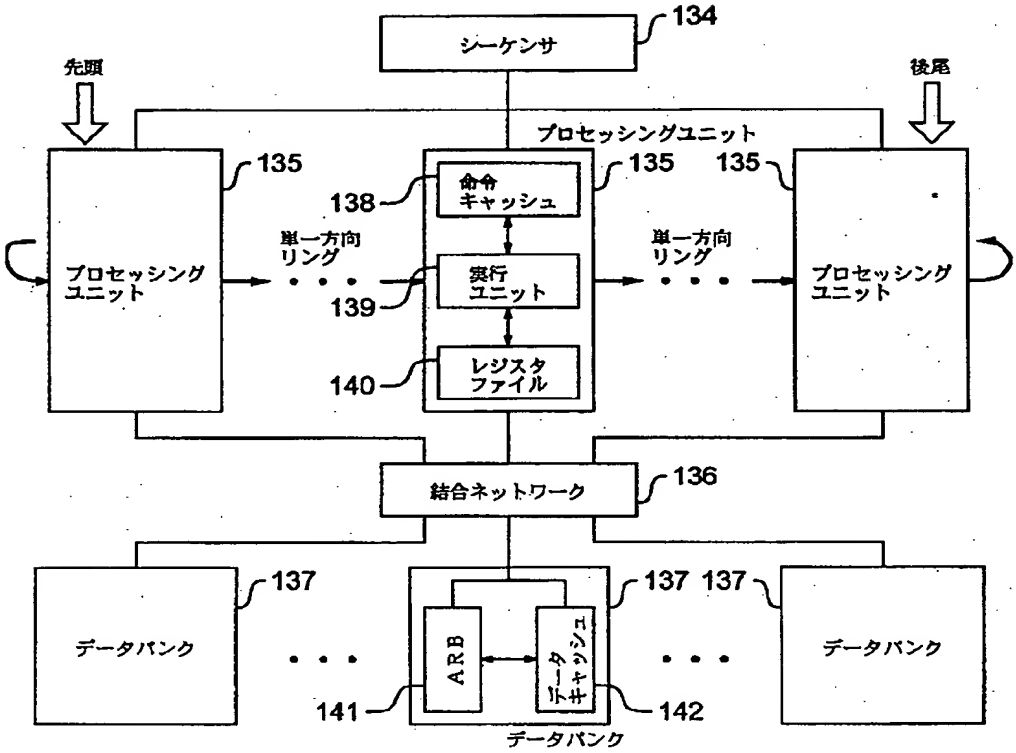
【図２９】


121
120
レジスタフリー
決定論理
状態テーブル
物理
＃０状態
＃１状態
使用
フリーレジスタ
検索論理
有効命令
順序バッファ
＃０より
ターゲット
物理レジスタ
旧対応番号
デコード時の
継承情報
有効命令
順序バッファ
＃１より
ターゲット
物理レジスタ
旧対応番号
デコード時の
継承情報
スレッド管理
ユニットより
現在の
継承情報
0
1
2
3
64
00
11
01
10
レジスタ
写像テーブル
＃０へ
新規確保
物理レジスタ
番号
新規確保
物理レジスタ
番号
レジスタ
写像テーブル
＃１へ
118
119

【図３２】


命令依存
解析装置
132
130
129
命令取得装置
命令解読装置
命令依存
解析装置
132
130
129
命令取得装置
命令解読装置
131
131
131
機能実行装置
命令調停装置
133

【図３３】


シーケンサ
134
先頭
後尾
プロセッシングユニット
135
135
135
138
命令
キャッシュ
単一方向
リング
実行
ユニット
139
単一方向
リング
プロセッシング
ユニット
プロセッシング
ユニット
140
レジスタ
ファイル
結合ネットワーク
136
データバンク
137
137
137
データバンク
ARB
データキャッシュ
141
142
データバンク